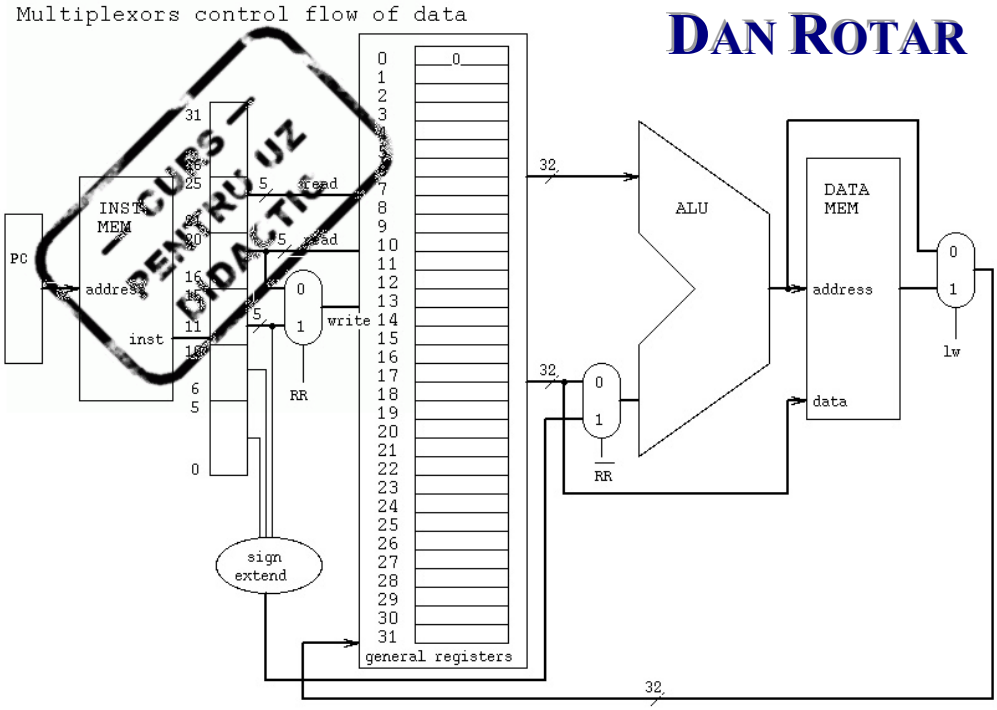
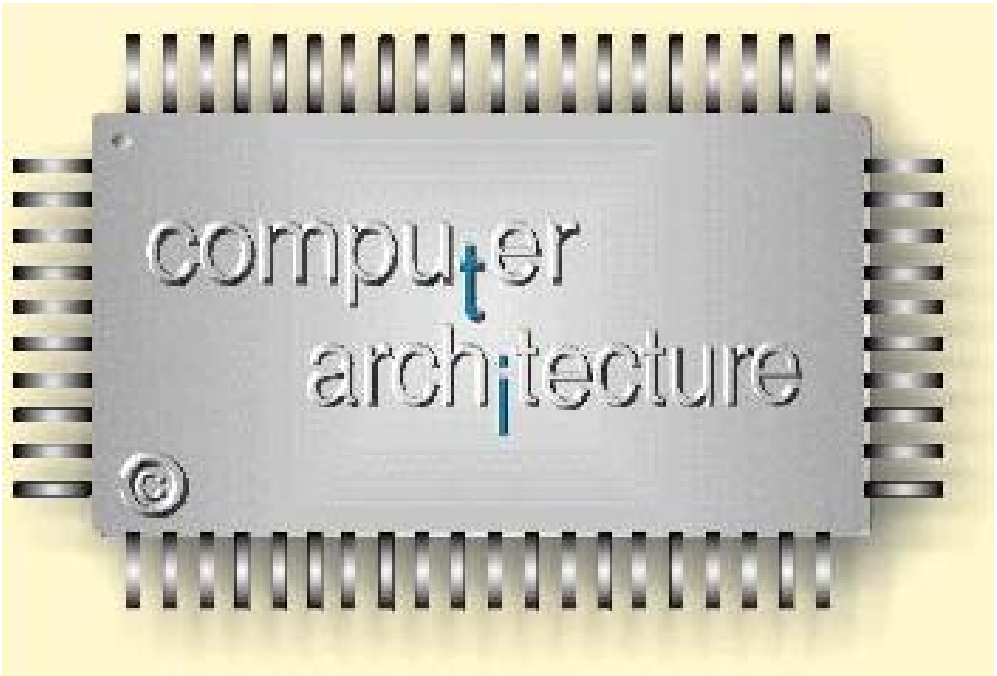
MINISTERUL EDUCAȚIEI ŞI CERCETĂRII
UNIVERSITATEA DIN BACĂU
FACULTATEA DE INGINERIE



CURS PENTRU UZ DIDACTIC

DAN ROTAR



EDITURA ALMA MATER

Cuprins

		pag.
	CAPITOLUL 1	5
	BAZE DE NUMERAȚIE	5
1.1.	Introducere	5
1.2.	Baza de numerație zece	6
1.3.	Baza de numerație doi	6
1.3.1.	Conversia binar-zecimală	7
1.3.2.	Conversia zecimal-binară	9
1.4.	Baza de numerație opt (sistemul octal)	11
1.5.	Baza de numerație şaisprezece (sistemul hexazecimal)	12
	CAPITOLUL 2	
	OPERAȚII ARITMETICE	13
2.1.	Introducere	13
2.2.	Operații aritmetice cu numere binare	13
2.2.1.	Adunarea	13
2.2.2.	Scăderea	14
2.2.3.	Înmulțirea	15
2.2.4.	Împărțirea întragă	16
2.3.	Operații aritmetice cu numere reprezentate în octal şi hexazecimal	17
2.3.1.	Adunarea şi scăderea în octal	18
2.3.2.	Adunarea şi scăderea în hexazecimal	19
	CAPITOLUL 3	
	ARHITECTURA SISTEMELOR DE CALCUL	20
3.1.	Introducere	20
3.2.	Arhitectura von Neumann	21
3.3.	Arhitectura Harvard	25
	CAPITOLUL 4	
	UNITATEA CENTRALĂ	28
	Introducere	28
4.2.	Microprocesorul universal (structura generală a unui microprocesor)	31

4.3. Caracteristicile principalelor tipuri de microprocesoare 32
4.3.1. Microprocesorul ZILOG Z80 32
4.3.2. Microprocesoarele INTEL 80x86 33
4.3.2.1. Microprocesorul INTEL 8086/8088 36
4.4. Procesoare de semnal digitale 50
4.4.1. Procesorul de semnal digital, TMS320F240 53
4.5. Microcalculatoare integrate, microcontrolere 57
4.5.1. Prezentare generală 57
4.5.2. Microcontrolerul AT90S2313 58
4.5.3. Microcontrolerul PIC 16F877 61

CAPITOLUL 5
MEMORIA 75
5.1. Prezentare generală 75
5.2. Aplicarea principiului "cache" în sistemele de calcul 81
5.2.1. Memoria Cache 81
5.2.2. Cache-ul de disc 84
5.2.3. Cache-ul microprocesorului 85
5.2.4. Cache-ul cu adresare directă (direct mapped) 86
5.2.5. Cache-ul cu adresare asociativă (fully associative) 86
5.2.6. Cache-ul parţial asociativ (set-associative) 87
5.3. Gruparea memoriilor 88
5.3.1. Creşterea capacităţii memoriei prin creşterea numărului de linii de date 90
5.3.2. Gruparea memoriilor pentru creşterea numărului de linii de adresă 91
5.3.3. Gruparea mixtă 93
5.4. Adresarea memoriilor 93
5.4.1. Adresarea absolută 93
5.4.2. Adresarea relativă (redundantă) 94

CAPITOLUL 6
PORTURI (INTERFEŢE)
6.1. Prezentare generală 95
6.2. Interfaţa serială programabilă 8251 98
6.3. Interfaţa logică programabilă 8255 106
6.4. Interfaţa USB (Universal Serial Bus) 114
6.4.1. Introducere 114
6.4.2. Prezentarea Universal Serial Bus 117
6.4.2.1. Vitezele USB 117
6.4.2.2. Conectorii 119
6.4.2.3. Caracteristici electrice 120
6.4.2.4. Identificarea vitezei 120
6.4.2.5. Alimentarea V_{BUS} 122
6.4.2.6. Protocolul 122
6.5. Interfeţele microcontrolerelor 124
6.5.1. Modulul convertor analog-digital (A/D) 125

6.5.1.1. Cerințele achziției analog-digitale 129
6.5.1.2. Selecția ceasului conversiei analog-digitale 130
6.5.1.3. Conversia A/D 131

CAPITOLUL 7
CIRCUITE SPECIALE
7.1. Introducere
7.2. Controlerul de întreruperi programabil 8259 132
7.3. Întreruperile microcontrolerului TMS320F240 138
7.3.1. Întreruperile managerului de evenimente (EV) 139
7.4. Accesul direct la memorie (DMA) 148
7.4.1. Circuitul 8257 pentru acces direct la memorie – DMA 150
7.4.1.1. Conexiunile externe 150
7.4.1.2. Registrele interne ale 8257 152
7.4.1.3. Registrele de canal 153
7.4.1.4. Registrul de mod 154
7.4.1.5. Registrul de stare 155
7.4.1.6. Efectuarea transferurilor cu DMA 8257 156
7.5. Circuitul contor/periodizator programabil 8253 158
7.6. Circuitele timer ale microcontrolerelor 164
7.6.1. Timerele de uz general GPTimer 164

Bibliografie 185

CAPITOLUL 1

BAZE DE NUMERAȚIE

1.1. Introducere

Valorile numerelor pot fi exprimate în diferite baze de numerație. Astfel, în activitățile umane obişnuite baza de numerație folosită este baza zece. Nu acelaşi lucru se întâmplă în sistemele numerice. Din considerente tehnologice, în sistemele numerice nu putem reprezenta decât două numere: zero şi unu. De regulă valoarea zero este asociată cu un nivel scăzut de tensiune iar valoarea unu este asociată cu un nivel ridicat de tensiune, diferența dintre tensiunea asociată valorii zero şi cea asociată valorii unu asigurând securitatea la perturbații a sistemului numeric (în sensul că dacă diferența între cele două tensiuni creşte, creşte şi imunitatea la perturbații a sistemului numeric).

O bază de numerație presupune existența unui număr de simboluri (numite adesea cifre) cu ajutorul cărora vor fi reprezentate numerele, un mod de scriere a numerelor şi o relație de calcul a valorii numărului. Numărul simbolurilor utilizate într-o bază de numerație defineşte numele bazei de numerație. Reprezentarea numerelor se face în prezent în exclusivitate prin scrierea poziționlă adică poziția numărului exprimă şi rangul acestuia (exponentul bazei de numerație). Simbolul (cifra) cel mai din dreapta reprezentării numărului are rangul minim (zero) iar simbolul (cifra) cel mai din stânga din reprezentarea numărului are rangul cel mai mare. Calculul valorii numărului se face prin înmulțirea valorii simbolului cu baza la puterea rangului. Atunci când se lucrează cu mai multe baze de numerație simultan este obligatorie indicarea bazei de numerație la fiecare număr scris. Acest lucru se face prin scrierea valorii bazei de numerație ca indice. De exemplu: 12_{10} sau 100110101_2 sau $1B2C_{16}$. Pentru indicarea bazei de numerație pot fi utilizate şi litere astfel:

z – (zecimal) pentru baza 10
b – (binar) pentru baza 2
o – (octal) pentru baza 8
h – (hexazecimal) pentru baza 16.

Exemplul de mai sus poate fi scris şi: : 12_z sau 100110101_b sau $1B2C_h$.

În continuare vor fi prezentate principalele baze de numerație utilizate în sistemele numerice.

1.2. Baza de numerație zece

Așa cum s-a arătat, baza de numerație zece, cea utilizată în activitățile umane, presupune existența a zece simboluri distincte pentru reprezentarea numerelor. Aceste simboluri sunt:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Reprezentarea numerelor este poziţională deci atunci când scriem un număr, poziția acestuia va reprezenta rangul său:

rang: 4 3 2 1 0
simbol: 7 5 2 1 3

Valoarea numărului reprezentat mai sus este: şapte zeci şi cinci de mii două sute treisprezece. Aceasta valoare a rezultat din regula prezentată mai sus adică:

valoare număr = $7 \times 10^4 + 5 \times 10^3 + 2 \times 10^2 + 1 \times 10^1 + 3 \times 10^0$

1.3. Baza de numerație doi

Numerele reprezentate în baza doi sunt numite în mod obişnuit numere binare. În baza doi sunt necesare numai două simboluri pentru reprezentarea numerelor şi acestea sunt: 0 (zero) şi 1 (unu). Datorită faptului că sistemele numerice lucrează, aşa cum s-a arătat mai sus, în sistem binar, studiul reprezentării numerelor în baza doi este important atunci când vorbim de tehnologia DSP. De asemenea prin asocierea unor valori de adevăr celor două simboluri: 0 – fals (false) şi 1 – adevărat (true) sistemele numerice vor putea lucra în logică binară sau Booleană. Abordarea logicii binare va fi făcută mai târziu în cuprinsul acestui manual.

TABELUL 1.1.

Număr binar	Număr zecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Rezenind la scrierea numerelor în baza doi trebuie spus faptul că toate regulile stabilite la baza zece se aplică şi aici. De exemplu, dacă vom scrie numărul binar:

rang 6 5 4 3 2 1 0
număr: 1 0 1 1 0 0 1

atunci valoarea acestuia va fi:

valoare = $1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 +$
$+ 0 \times 2^1 + 1 \times 2^0 = 64_z + 16_z + 8_z + 1_z = 89_z$

În tabelul 2.1 este prezentată echivalența primelor

16 numere binare cu cele zecimale. Pentru acest tabel putem face următoarele observaţii:

- daca citim numerele binare din acest tabel pe coloană observăm faptul că succesiunea cifrelor zero şi unu depinde de rangul cifrei. Astfel la rangul zero întâlnim succesiunea 0,1,0,1,0,1, ... , la rangul unu succesiunea 0,0,1,1,0,0,1,1, ... , şi aşa mai departe;
- un număr zecimal putere a lui doi va determina o cifră binară cu un singur simbol unu pe poziţia exponentului puterii lui doi.

Este evident faptul ca atunci când se doreşte conversia numerelor dintr-o bază în alta este dificil de utilizat tabele, în special atunci când se lucrează cu numere mari. Din acest motiv se stabilesc anumiţi algoritmi de conversie pentru simplificarea translatării numerelor dintr-o bază în alta.

1.3.1. Conversia binar-zecimală

Conversia unui numar binar întreg în echivalentul său zecimal, cunoscând că operaţiile se efectuează în sistemul zecimal, se face cu ajutorul metodei înmulţirii repetate cu 2. Pentru a justifica, se consideră numărul binar ***P*** exprimat prin:

$$P = b_{n-1}\,2^{n-1} + b_{n-2}\,2^{n-2} + ... + b_1\,2^1 + b_0\,2^0$$

sau:

$$P = (\{[\overbrace{(b_{n-1}\,2 + b_{n-2})}^{P_1}\,2 + b_{n-3}]\,2 + b_{n-4}\}\,2 + ... + b_1)\,2 + b_0$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{P_2}$$

relaţie din care rezultă algoritmul conversiei numerelor întregi binare în numere zecimale:

- pentru a forma mărimea intermediară P_1 se înmulţeşte cu 2 cifra cea mai semnificativă (cifra de rang maxim) a numărului binar, adăugând următoarea cifră semnificativă;
- pentru a obţine mărimea intermediară P_2, se înmulţeşte P_1 cu 2 adăugând cea de-a treia cifră semnificativă;
- se continuă acest algoritm până la adăugarea cifrei cel mai puţin semnificative a numărului binar (cea de rang minim);
- numărul zecimal obţinut este echivalentul zecimal al numărului binar dat.

EXEMPLU

Să se transforme numărul binar 10110110111 în echivalentul său zecimal:

1 x 2 + 0 = 2	$P_1 = 2$
2 x 2 + 1 = 5	$P_2 = 5$
5 x 2 + 1 = 11	$P_3 = 11$
11 x 2 + 0 = 22	$P_4 = 22$
22 x 2 + 1 = 45	$P_5 = 45$
45 x 2 + 1 = 91	$P_6 = 91$
91 x 2 + 0 = 182	$P_7 = 182$
182 x 2 + 1 = 365	$P_8 = 365$
365 x 2 + 1 = 731	$P_9 = 731$
731 x 2 + 1 = 1463	$P_8 = 1463$

deci: $10110110111_B = 1463_Z$

Pentru conversia binar-zecimală a unui număr binar fracționar se folosește metoda împărțirii repetate cu 2. Pentru a justifica, se consideră numărul binar fracționar ***Q*** dat de:

$$Q = b_{-1}\,2^{-1} + b_{-2}\,2^{-2} + ... + b_{-m}\,2^{-m}$$

care poate fi pus şi sub forma:

$$Q = 2^{-1}\,(b_{-1} + 2^{-1}\,\{\,b_{-2} + 2^{-1}\,[b_{-3} + ... + 2^{-1}\,\overbrace{(b_{-m+1} + 2^{-1}b_{-m})}^{Q_1}]\})$$

de unde rezultă algoritmul conversiei numerelor binare fracționare în numere zecimale:

- pentru a obține mărimea intermediară Q_1 se împarte cifra cea mai puțin semnificativă (de rang minim) cu 2, adăugând urmatoarea cifră semnificativă;
- pentru a obține mărimea intermediară Q_2 se împarte Q_1 cu 2 adăugând a treia – a treia cifră de la dreapta spre stânga – cifră semnificativă;
- se continuă acest algoritm până când împărțirea care corespunde cifrei 0 de la stânga virgulei a fost efectuată;
- numărul obținut este echivalentul zecimal al numărului binar dat.

EXEMPLU

Să se transforme numărul binar fracționar 0,001101111 în echivalentul său zecimal:

1	: 2 + 1 = 1,5	$Q_1 = 1,5$
1,5	: 2 + 1 = 1,75	$Q_2 = 1,75$
1,75	: 2 + 1 = 1,875	$Q_3 = 1,875$
1,875	: 2 + 0 = 0,9375	$Q_4 = 0,9375$
0,9375	: 2 + 1 = 1,46875	$Q_5 = 1,46875$
1,46875	: 2 + 1 = 1,734375	$Q_6 = 1,734375$
1,734375	: 2 + 0 = 0,8671875	$Q_7 = 0,8671875$

0,8671875	: 2 + 0 = 0.4335937	$Q_8 = 0,4335937$
0,4335937	: 2 + 0 = 0,2167968	$Q_9 = 0,2167968$

deci: $0,001101111_B = 0,2167968_Z$

Dacă se cere conversia unui număr binar care are atât parte întreagă cât şi parte fracționară în echivalentul său zecimal, se aplică părții întregi algoritmul corespunzător conversiei numerelor întregi, iar părții fracționare algoritmul corespunzător părții fracționare.

1.3.2. Conversia zecimal-binară

Conversia zecimal-binară a numerelor întregi se face după metoda împărțirii repetate prin 2. Pentru a justifica aceasta, se consideră numărul intreg ***N*** în baza 10, care poate fi exprimat în funcție de puterile lui 2 sub următoarea formă :

$$N = a_{n-1}\, 2^{n-1} + a_{n-2}\, 2^{n-2} + ... + a_1\, 2^1 + a_0\, 2^0$$

sau:

$$N = (a_{n-1}\, 2^{n-2} + a_{n-2}\, 2^{n-3} + ... + a_1)\ 2 + a_0 = N_1\, 2 + a_0$$

Din ultima egalitate rezultă că cifra cea mai semnificativă a_0 din reprezentarea binară a numărului N constituie restul împărțirii lui N cu 2.
În mod analog, numărul N_1 se poate exprima prin :

$$N = (a_{n-1}\, 2^{n-3} + a_{n-2}\, 2^{n-4} + ... + a_2)\ 2 + a_1 = N_2\, 2 + a_1$$

Adică a_1, care reprezintă cifra semnificativă de rang imediat următor în reprezentare binară, constituie restul împărțirii lui N_1 cu 2.

De aici rezultă algoritmul conversiei numerelor întregi zecimale în numere binare:

- se împarte numărul intreg în baza zece N prin 2; se obține câtul N_1 şi restul a_0;
- se împarte câtul N_1 prin 2; se obține câtul N_2 şi restul a_1;
- se continuă această operație până se ajunge la un cât N_n egal cu zero;
- resturile obținute sunt cifrele numărului binar (biții), a_0 fiind cifra cea mai puțin semnificativă, a_1 cifra următoare ş. a. m. d.

EXEMPLU

Să se transforme numărul zecimal 53 în echivalentul său binar.

53 : 2 = 26 + 1	$a_0 = 1$
26 : 2 = 13 + 0	$a_1 = 0$

13 : 2 = 6 + 1	$a_2 = 1$
6 : 2 = 3 + 0	$a_3 = 0$
3 : 2 = 1 + 1	$a_4 = 1$
1 : 2 = 0 + 1	$a_5 = 1$

Deci $53_{10} = 110101_2$

Folosind acelaşi algoritm se prezintă un alt mod de aranjare a calculelor, în aşa fel, ca la sfârşitul operaţiei să se obţină direct numărul binar fără a fi necesară rescrierea sa. În acest sens, se aşează succesiunea calculelor după schema prezentată mai jos, unde al *n*-lea cât este ultimul cât, egal cu zero :

Cât n ← Cât (n –1) ← Cât (n –2) ← .... Cât 2 ← Cât 1 ← Număr
Rest n Rest (n-1) .... Rest 3 Rest 2 Rest 1

Considerând acelaşi exemplu se obţine :

0	1	3	6	13	26	53
	1	1	0	1	0	1

Conversia zecimal-binară a numerelor fracţionare se face dup metoda înmulţirii repetate cu 2. Pentru a justifica, se consideră numărul fracţionar zecimal ***M***, care poate fi exprimat în funcţie de puterile numărului 2 prin :

$$M = a_{-1} 2^{-1} + a_{-2} 2^{-2} + ... + a_{-m} 2^{-m}$$

Prin înmulţirea ambelor părţi ale ecuaţiei de mai sus cu 2, se obţine :

$$2 M = a_{-1} + (a_{n-2} 2^{-1} + ... + a_{-m} 2^{-m+1}) = a_{-1} + M_1$$

partea dreaptă a egalităţii fiind formată din numărul întreg a_{-1}, care reprezintă cifra cea mai semnificativă a numărului binar subunitar şi fracţia M_1. Aplicând acelaşi procedeu lui M_1, rezultă:

$$2 M_1 = a_{-2} + (a_{n-3} 2^{-1} + ... + a_{-m} 2^{-m+2}) = a_{-2} + M_2$$

adică se obţine următoarea cifră semnificativă a_{-2} a numărului fracţionar binar şi fracţia M_2.

Cu aceasta, algoritmul conversiei numerelor fracţionare zecimale în numere fracţionare binare este următorul:

- se înmulţeşte numărul zecimal fracţionar M cu 2; rezultă bitul a_{-1} şi partea fracţionară M_1;
- se înmulţeşte partea fracţionară M_1 cu 2; rezultă bitul a_{-2} şi partea fracţionară M_2;

- se contiună acestă operație până când M_m devine egal cu zero sau ne limităm la un număr de cifre binare în funcție de precizia impusă.

EXEMPLU

1. Să se transforme numărul 0,40625 în echivalentul său binar.

0,40625 x 2 = 0,81250	$a_{-1} = 0$
0,8125 x 2 = 1,6250	$a_{-2} = 1$
0,625 x 2 = 1,250	$a_{-3} = 1$
0,25 x 2 = 0,5	$a_{-4} = 0$
0,5 x 2 = 1,0	$a_{-5} = 1$

deci $0,40625_z = 0,01101_b$

2. Să se transforme numărul zecimal 0,7 în echivalentul său binar.

0,7 x 2 = 1,4	$a_{-1} = 1$
0,4 x 2 = 0,8	$a_{-2} = 0$
0,8 x 2 = 1,6	$a_{-3} = 1$
0,6 x 2 = 1,2	$a_{-4} = 1$
0,2 x 2 = 0,4	$a_{-5} = 0$
0,4 x 2 = 0,8	$a_{-6} = 0$
.....	

deci $0,7_{10} = 0,101100...._2$

1.4. Baza de numerație opt (sistemul octal)

Utilizatorii primei generații de calculatoare au întâmpinat dificultăți în manevrarea cifrelor binare, întrucât un număr exprimat în sistemul binar are o lungime de circa trei ori mai mare decât în exprimarea zecimală. Din acest motiv, s-a căutat un sistem de numerație mai apropiat de baza zece, cu o contingență directă cu sistemul binar. Cunoscând că 8 este o putere întreagă a lui 2 ($2^3 = 8$), aceste calități sunt întrunite de sistemul de numerație octal, care are opt cifre pentru exprimare, de la 0 la 7 inclusiv. Ca urmare, conversia binar octală şi octal binară se poate face direct, în primul caz înlocuind un grup de trei cifre binare socotite de la dreapta şi de la stânga virgulei prin echivalentul lor octal, iar în al doilea caz înlocuind fiecare cifră octală printr-un grul de trei cifre binare.

EXEMPLU

1. Să se transforme numărul binar 10111,00 11 în echivalentul său octal.

$$\underbrace{010}_{2}\ \underbrace{111}_{7},\ \underbrace{001}_{1}\ \underbrace{100}_{4}{}_2 = 27,14_8$$

2. Să se transforme numărul octal 26,341 în echivalentul său binar.

$$26{,}341_8 = \underbrace{010}_{2}\ \underbrace{110}_{6},\ \underbrace{011}_{3}\ \underbrace{100}_{4}\ \underbrace{001}_{1}{}_2$$

Prin folosirea sistemului octal, calculele binare sau structura informației din interiorul calculatorului nu se schimbă, structură care poate fi verificată cu uşurință din exterior. Din acest motiv, calculatoarele din prima şi a doua generație aveau o lungime a cuvântului egală cu un multiplu a lui 3, obişnuit 24, 36 sau 42 de biți, pentru a facilita trecerea din sistemul octal în binar sau invers.

Se menționează că pentru transformarea numerelor din baza de numerație 10 în baza de numerație 8 sau invers, cu calculele făcute în baza 10, toți algoritmii deduşi anterior rămân valabili, cu observația că înmulțirile, respectiv împărțirile prin 2 se înlocuiesc cu înmulțiri, respectiv împărțiri, cu 8.

1.5. Baza de numerație şaisprezece (sistemul hexazecimal)

Pentru a reprezenta în calculator alte caractere decât numere, adică litere, semne de punctuație etc. (caractere alfanumerice), au fost utilizate inițial 6 cifre binare. În cursul operațiilor de citire sau imprimare, dispozitivul respectiv făcea automat transformarea fiecărui caracter în cele 6 cifre binare sau invers.

Folosind 6 cifre binare, adică două cifre octale, nu puteau fi reprezentate decât 64 de caractere distincte. Acest număr de caractere s-a dovedit a fi în scurt timp insuficient pentru aplicațiile practice şi din acest motiv sa-a trecut, la majoritatea calculatoarelor generației a treia, la alocarea a 8 cifre binare, adică două hexazecimale, fiecărui caracter. Un grup de 8 cifre binare este cunoscut şi sub denumirea de *octet* sau *bait* şi reprezintă cea mai mică diviziune a informației care poate fi prelucrată în aceste calculatoare.

Întrucât baza de numerație 16 este mai mare ca baza de numerație 10, cifrele mai mari de 9 sunt notate în ordine cu literele alfabetului latin, adică :

A – zece
B – unsprezece
C – doisprezece
D – treisprezece
E – paisprezece
F – cincisprezece.

Pentru conversia din baza 10 în 16 sau invers, sunt utilizate aceleaşi metode, făcându-se calculele în baza 10.

Conversia binar-hexazecimală şi hexazecimal-binară se face în mod similar celei binar-octale, respectiv octal-binare, cu observația că se vor lua grupe de câte 4 cifre binare în loc de 3.

CAPITOLUL 2

OPERAȚII ARITMETICE

2.1. Introducere

În acest capitol se vor prezenta operațiile aritmetice simple realizate cu numere binare (numere reprezentate în baza doi), numere octale (numere reprezentate în baza opt) sau hexazecimale (numere reprezentate în baza saisprezece).

Pentru exemplificarea operațiilor efectuate se vor folosi numai numere întregi pozitive (numere naturale) deoarece acesta este modul de bază al reprezentării informației într-un sistem de calcul numeric.

2.2. Operații aritmetice cu numere binare

2.2.1. Adunarea

Operația de adunare poate fi descrisă cu ajutorul celor patru combinații posibile între două numere binare:

0 + 0 = 0; transport la rangul superior: 0
0 + 1 = 1; transport la rangul superior: 0
1 + 0 = 1; transport la rangul superior: 0
1 + 1 = 0; transport la rangul superior: 1

În mod obişnuit se scrie: $1_b + 1_b = 10_b$ deoarece în această situație avem un transport la rangul superior. Este evident faptul că: $10_b = 2_z$.

EXEMPLE

1) Să se adune numărul binar 110111 cu numărul binar 101.

```
transport      111
             110111 +
                101
            ------
             111100
```

putem efectua verificarea in zecimal: $110111_b = 55_z$, $101_b = 5_z$, $111100_b = 60_z$.

Verificarea este imediată. Se observă că dacă trebuie să adunăm un sir de ***n*** cifre unu în binar, rezultatul pe rangul respectiv va fi zero dacă numărul de cifre unu adunate este par sau unu dacă numărul de cifre adunate este impar iar transportul se face peste

n:2 (împărțire întreagă) ranguri pornind de la rangul respectiv (cel la care se efectueaza adunarea).

2) Să se adune numărul binar 11111 cu numărul binar 1.

```
transport      1111
               11111 +
                   1
              ------
              100000
```

De fapt rezultatul poate fi văzut imediat dacă ne uităm la tabelul 1.1. Dupa un numar cu o succesiune de cifre unu urmează un număr binar care are o cifră unu urmat de o succesiune de cifre zero. Numărul cifrelor zero este egal cu numărul cifrelor unu al numărului care se adună. De exemplu $1111_b + 1_b = 10000_b$.

2.2.2. Scăderea

Operația de scădere poate fi descrisă cu ajutorul celor patru combinații posibile între două numere binare:

0 – 0 = 0; împrumut de la rangul superior: 0
0 – 1 = 1; împrumut de la rangul superior: 1
1 – 0 = 1; împrumut de la rangul superior: 0
1 – 1 = 0; împrumut de la rangul superior: 0

În mod obişnuit se scrie: $10_b – 1_b = 1_b$ deoarece în această situație avem un împrumut de la rangul superior. Este evident faptul că: $10_b = 2_z$.

EXEMPLE

1) Să se efectueze scăderea numerelor binare: 11001 şi 11.

```
împrumut       11
               11001 –
                  11
              ------
               10110
```

Verificarea in zecimal: $11001_b = 25_z$, $11_b = 3_z$, $10110_b = 22_z$. $25_z – 3_z = 22_z$.

2) Să se efectueze scăderea 11001 – 1011.

```
împrumut       111
               11001 –
                1011
              ------
               01110
```

Verificarea in zecimal: $11001_b = 25_z$, $1011_b = 11_z$, $1110_b = 14_z$. $25_z – 11_z = 14_z$.

Se pot face observații similare cu cele de la adunare.

3) Sa se scadă din 1000_b valoarea 1. Răspunsul este imediat: 111_b.

2.2.3. Înmulțirea

Operația de înmulțire poate fi descrisă cu ajutorul celor patru combinații posibile între două numere binare:

0 x 0 = 0; 0 x 1 = 0; 1 x 0 = 0; 1 x 1 = 1.

Operația de înmulțire se efectuează identic cu cea din baza zece. Vom analiza în continuare câteva exemple.

EXEMPLE

1) Să se efectueze înmulțirea în binar: 110101 x 1101.

```
  110101 x
    1101
  ------
  110101
 000000
110101
110101
----------
1010110001
```

Verificarea în zecimal: $110101_b = 53_z$, $1101_b = 13_z$, $1010110001_b = 689_z$. Rezultă 53_z x $13_z = 689_z$.

Din acest exemplu se observă că operația de înmulțire poate fi înlocuită cu operatii de deplasare a deînmulțitului spre stânga şi adunarea rezultatelor obținute. Pentru aceasta se parcurge înmulțitorul de la dreapta la stânga cifră cu cifră. Pentru fiecare cifră unu a înmulțitorului se scrie deînmulțitul deplasat spre stânga cu un număr de cifre egal cu rangul cifrei unu a înmulțitorului. La final se sumează rezultatele obținute.

2) Să se efectueze înmulțirea numărului 101101 cu numărul 1101.
Vom aplica metoda deplasării spre stânga a deînmulțitului şi adunarea şirurilor obținute.

deînmulțitul deplasat la stânga cu 0 paşi	101101
deînmulțitul deplasat la stânga cu 2 paşi	101101
deînmulțitul deplasat la stânga cu 3 paşi	101101
rezultatul (suma)	1001001001

Verificarea în zecimal: $101101_b = 45_z$, $1101_b = 13_z$, $1001001001_b = 585_z$.
Rezultă: 45_z x $13_z = 585_z$.

2.2.4. Împărțirea întreagă

Operația de împărțire poate fi descrisă cu ajutorul celor patru combinații posibile între două numere binare:

0 : 0 = 0; 0 : 1 = 0; 1 : 0 = imposibil; 1 : 1 = 1.

Operația de împărțire se efectuează identic cu cea din baza zece. Vom analiza în continuare câteva exemple.

EXEMPLE

1) Să se efectueze împărțirea numărului 11010110101 cu 1011.

```
11010110101|1011
 1011      10011100
 0010011
   1011
   10000
    1011
    001011
      1011
      000001
```

Rezultatul este: 10011100 rest 1. Verificarea în zecimal: $11010110101_b = 1717_z$, $1011_b = 11_z$, $10011100_b = 156_z$. $1717_z : 11_z = 156_z$ rest 1.

Din acest exemplu se observă faptul că operația de împărțire întreagă se reduce la operații succesive de deplasare la dreapta a împărțitorului şi scăderi din deîmpărțit, începând de la cifrele de rang maxim ale deîmpărțitului. Se contiunuă cu deîmpărțitul nemodificat dacă rezultatul scăderii este negativ sau cu rezultatul scăderii dintre deîmpărțit şi împărțitor dacă rezultatul scăderii este zero sau pozitiv. La fiecare scădere se notează cifra cîtului care este unu dacă rezultatul scăderii este zero sau pozitiv (scăderea se poate efectua) sau zero dacă rezultatul scăderii este negativ (scăderea nu se poate efectua). Operația de scădere se opreşte când se efectuează o scădere din bitul de rang zero al deîmpărțitului. Valoarea deîmpărțitului rămasă constituie restul împărțirii.

2) Să se efectueze împărțirea dintre numerele 111101101 si 101.
Vom apela la procedeul deplasării la dreapta a împărțitorului şi scăderea din deîmpărțit.

```
111101101
101
010
```

- rezultatul scăderii este pozitiv deci se continuă cu valoarea rămasă în urma scăderii şi se face o deplasare la dreapta; cît = 1

010101101
 101
0000 - rezultatul scăderii este zero deci se continuă cu valoarea rămasă în urma scăderii şi se face o deplasare la dreapta; cît = 11

000001101
 101 - rezultatul scăderii este negativ deci nu se efectuează scăderea şi se face deplasarea dreapta a împărțitorului; cît = 110

000001101
 101 - rezultatul scăderii este negativ deci nu se efectuează scăderea şi se face deplasarea dreapta a împărțitorului; cît = 1100

000001101
 101 - rezultatul scăderii este negativ deci nu se efectuează scăderea şi se face deplasarea dreapta a împărțitorului; cît = 11000

000001101
 101
 001 - rezultatul scăderii este pozitiv deci se continuă cu valoarea rămasă în urma scăderii şi se face o deplasare la dreapta; cît = 110001

000000011
 101 - rezultatul scăderii este negativ deci nu se efectuează scăderea; cît = 1100010 iar restul este 11 deoarece s-a incercat scăderea din cifra de rang zero a deîmparțitului.

Verificarea în zecimal: $111101101_b = 493_z$, $101_b = 5_z$, $1100010_b = 98_z$, $11_b = 3_z$.
$493_z : 5_z = 98_z$ rest 3_z.

2.3. Operații aritmetice cu numere reprezentate în octal şi hexazecimal

Reprezentarea numerelor în octal sau hexazecimal se face, asa cum s-a arătat, în scopul cresterii clarității reprezentării valorilor numerice. Trebuie subliniat aici, încă o dată, că în sistemele numerice de calcul, singurul mod de reprezentare a informației este cel binar. Utilizarea altor baze de numerație cum sunt cel octal sau cel hexazecimal se face doar în scopul creşterii clarității documentației însoțitoare. Baza de numerație opt şi baza de numerație şaisprezece au avantajul că au baze puteri a lui doi şi din acest motiv permit scrierea condensată şi comodă a numerelor binare.

Toate operațiile efectuate în altă bază de numerație decât baza zece se bazează pe principii similare, aşa cum s-a văzut şi la descrierea operațiilor aritmetice în baza doi.

Este important de reținut faptul că atunci când rezultatul adunării depăşeşte ca valoare valoarea bazei se produce un transport în rangurile superioare iar cînd se efectuează scăderea unui număr mai mare dintr-un număr mai mic se produce un împrumut de la rangurile superioare (identic cu ceea ce deja ştim de la operațiile în baza zece).

Cele mai importante operații efectuate în octal sau în hexazecimal sunt cele de adunare şi de scădere. Din acest motiv vom exemplifica pe scurt, în continuare modul în care se efectuează aceste operații în cele două baze de numerație.

2.3.1. Adunarea şi scăderea în octal

Adunarea, exemple

1) Să se adune numerele reprezentate în baza opt 723_o cu 523_o.

transport 1
723_o +
523_o
1446_o

În rangul doi, la adunarea cifrelor 7_o cu 5_o se produce un transport deoarece $7_z + 5_z = 12_z$ este o valoare mai mare decât valoarea bazei care este 8. În acest caz se procedează astfel: se scade din valoarea obținută valoarea bazei $12_z - 8_z = 4_z$, valoarea respectivă se scrie pe rangul doi şi avem un transport la rangul 3.
Trebuie ținut cont de faptul că în octal: $7_o + 1_o = 10_o$.

2) Să se efectueze adunarea în octal $375_o + 276_o$.

transport 11
375_o +
276_o
673_o

Scăderea, exemple

1) Să se efectueze scăderea în octal: $532_o - 251_o$.

împrumut 1
532_o –
251_o
261_o

La scăderea de pe rangul unu: $3_o - 5_o$, trebuie să se efectueze un împrumut din rangul superior. O cifră împrumutată din rangul superior înseamna opt unități în rangul inferior. Deci, în urma împrumutului, avem: $8_z + 3_z - 5_z = 6_z$.

2) Să se scadă în octal din $453_o - 264_o$.

împrumut 11
453_o –
264_o
167_o

Vrificarea în zecimal este imediată dacă se face conversia din baza opt în baza zece utilizând relația:

valoare zecimală = $o_{n-1}\,8^{n-1} + o_{n-2}\,8^{n-2} + ... + o_1\,8^1 + o_0\,8^0$

unde: o_0, o_1, o_2, …, o_n sunt cifrele numărului în baza opt.

2.3.2. Adunarea şi scăderea în hexazecimal

Având în vedere faptul că operațiile sunt similare cu cele din octal, cu excepția faptului că baza are valoarea şaisprezece, vom prezenta numai câte un exemplu pentru adunare şi unul pentru scădere.

Adunare, exemplu

Sa se adune numerele în hexazecimal: $2A57_h + 57B9_h$

transport	111
	$2A57_h$ +
	$57B9_h$
	8210_h

În acest caz depăşirea se produce când suma este mai mare decât şaisprezece. De exemplu, $7_z + 9_z = 16_z = 10_h$

Scădere, exemplu

Să se efecueze scăderea: $5C2B_h - 3ACF_h$.

împrumut	11
	$5C2B_h$ –
	$3ACF_h$
	$215C_h$

În cazul împrumutului, de pe rangul superior se împrumuta şaisprezece unități. De exemplu, în cazul $B_h - F_h$, avem un împrumut şi se obține: $16_z + 11_z$ (B_h) – 15_z (F_h) = 12_z (C_h).

CAPITOLUL 3

ARHITECTURA SISTEMELOR DE CALCUL

3.1. Introducere

Un sistem de calcul reprezintă o maşină automată destinată prelucrării informaţiilor. O astfel de maşină va interacţiona cu mediul extern pentru preluarea informaţiilor de intrare şi furnizarea informaţiilor rezultate în urma prelucrării informaţiei de intrare. Prelucrarea informaţiei se face într-un anumit scop ceea ce determină o anumită activitate internă a sistemului de calcul. În acest moment putem distinge mai multe faze ale prelucrării informaţiei de către un sistem de calcul.

În primul rând informaţia prezentată la intrarea sistemului de calcul reprezintă în, accepţia generală, un semnal. Acest semnal are un suport fizic şi este reprezentat de o mărime electrică, mecanică, optică etc. O astfel de mărime este descrisă de anumiti parametrii măsurabili şi are un suport energetic care permite interacţiunea cu această mărime. O astfel de mărime trebuie să fie transformată (adaptată) în aşa fel încât aceasta să fie compatibilă cu formatul de intrare acceptat de sistemul de calcul.

În al doilea rând informaţia o data prelucrată de sistemul de calcul trebuie furnizată mediului extern într-un anumit format impus de aplicaţia pentru care se efectuează prelucrarea informaţiei. Este limpede că va fi necesară o nouă trensformare a informatiei din formatul furnizat la ieşire de către sistemul de calcul în formanul impus de aplicaţie.

În sfârşit, în afară de cele două faze ce implică transformări ale suportului informaţiei, putem distinge şi a treia fază, cea a prelucrării interne, în sistemul de calcul a informaţiei.

Pentru a asigura o flexibilitate cât mai mare a unui astfel de sistem, cele trei faze ale prelucrării informaţiei vor fi asigurate de către elemente diferite. Astfel, o anumită modificare ce s-ar impune pe parcurs nu va afecta deât un anumit element din structura sistemului.

În prezent, cele trei faze ale prelucrării informaţiei, sunt realizate astfel:

- transformarea semnalelor într-un format compatibil cu cel acceptat de intrarea sistemului de calcul şi transformarea informaţiei furnizate de sistemul de calcul în semnale necesare aplicaţiei, este realizată de către dispozitivele periferice. Acestea, deşi sunt strâns legate de funcţionarea sistemului de calcul sunt considerate elemente externe sistemului de calcul;
- prelucrarea internă în sistemul de calcul a informaţiei va fi faza de prelucrare a informaţiei care determină o anumită structură a sistemului de calcul. Această structură va fi completată cu elemente de legătură cu dispozitivele periferice, elemente ce poartă denumirea generică de *interfeţe*.

Pentru început ne vom ocupa de structura internă a sistemului de calcul ce constituie arhitectura internă a acestuia după care se vor discuta câteva aspecte legate de periferice întregul reprezentând arhitectura unui sistem de calcul.

Din punct de vedere al tipului informației prelucrate în sistemul de calcul, aceste pot fi de două tipuri:

- *calculatoarele analogice* care prelucrează semnale continui în timp, adică semnale ce sunt descrise de o funcție continuă de timp;
- *calculatoare numerice* ce prelucrează numere, adică informația prezentată în format numeric.

Între aceste două tipuri de calculatoare există deosebiri esențiale deşi amândouă sunt destinate prelucrării informației. Primul aspect se referă la tipul informației prelucrate. Dacă la calculatoarele analogice informația trebuie să aibă un anumit nivel al suportului energetic pentru ca aceasta să poată fi prelucrată, la calculatoarele numerice informația este lipsită de suportul energetic fiind reprezentată de elemente abstracte cum sunt numerele. Acest lucru aduce o serie de avantaje importante în favoarea calculatoarelor numerice ceea ce face ca utilizarea acestora să fie din ce în ce mai răspândită.

Pentru a lămuri lucrurile vom considera un exemplu simplu. Un instrument de măsură a tensiunii (voltmetru) poate fi considerat ca un sistem de calcul foarte simplu. Un astfel de instrument preia o informație dein mediul extern (tensiunea electrică), o adaptează la formatul acceptat la intrare (de regulă printr-un divizor de tensiune), prelucrează informația (prin comparare cu un element etalon) şi furnizează la ieşire o informație compatibilă cu aplicația (în cazul cel mai simplu afişând rezultatul măsurătorii într-un anumit fel). În cazul unui voltmetru analogic (cu instrument de măsură) informația de intrare trebuie să furnizeze o anumită energie pentru ca măsurătoarea să poată fi efectuată (această energie preluată de la semnalul măsurat va determina o deplasare a acului indicator proporțională cu mărimea măsurată). În cazul unui voltmetru numeric prelucrarea informației se face în principal prin transformarea acesteia într-o valoare numerică. Din acest motiv energia preluată de la semnalul de intrare va fi cu mult mai mică (această energie nemaijucând nici un rol în prelucrarea semnalelor). Vom aminti numai două din concluziile ce pot fi desprinse din acest exemplu: măsurătoarea efectuată cu un instrument numeric este mai precisă prin scăderea cantității de energie preluate de la semnalul de intrare (cunoscându-se faptul că preluarea unei anumite cantități de energie din semnalul de măsurat duce la alterarea acestuia) şi mărimile reprezentate prin numere cum sunt cele de la voltmetrul numeric nu pot fi influențate de condițiile de mediu (temperatură, presiune, umiditate etc) cum sunt cele analogice prezente în cazul voltmetrului analogic.

Aşa cum s-a afirmat, în interiorul sistemului de calcul se realizează o anumită prelucrare a informației. Este important de văzut în ce fel anume se realizează această prelucrare. Dacă ne gândim la exemplul de mai sus este limpede că voltmetrul analogic îndeplineşte o anumită funcție prin modul în care este construit adică prin conexiunile electrice existente între elementele componente. Un astefel de sistem se numeşte cu *logică cablată* adică modul de prelucrare a informației este determinat de modul

perticular de conectare a elementelor componente. Acelaşi lucru se poate realiza şi în cazul unui sistem numeric şi deci şi astfel de sisteme pot fi în logică cablată. Un sistem realizat în logică cablată nu va putea îndeplini decât o anumită funcţie (în exemplul nostru, nu ne vom putea gândi să folosim voltmetrul la altceva decât la măsurarea tensiunii) cea pentru care s-a realizat un anumit mod de conectare între elementele componente. Rezultă că un sistem în logică cablată este lipsit de flexibilitate, modificarea funcţiei îndeplinite presupunând (cel puţin) modificarea conexiunilor existente şi (eventual) necesitatea adăugării unor componente noi. Un astfel de sistem are totuşi avantajul simplităţii în sensul că el nu va conţine decât elementele componente strict necesare apicaţiei particulare pentru care este destinat.

Sistemele numerice prin faptul că prelucrează informaţia sub formă numerică permit realizarea unor structuri generale, independente de aplicaţia pentru care vor fi folosite. În această situaţie prelucrarea internă a informaţiei nu mai este legată de coneziunile existente între elementele structurii fizice. Pe această structură fizică generică (independentă de aplicaţie) prelucrarea informaţiei se va face pe baza unui program (listă de comenzi) ce se va afla în *memoria* sistemului de calcul. Astfel de sisteme se numesc *cu logică* programată. Ele sunt flexibile (schimbarea programului înseamnă schimbarea sau modificarea aplicaţiei pentru care sunt destinate) dar şi redundante în sensul că structura fizică generală poate avea mai multe elemente decât sunt necesare pentru o anumită aplicaţie.

Calculatoarele analogice nu pot fi construite decât în logică cablată pe când cele numerice pot fi construite atât în logică cablată cât şi în logică programată. Trebuie precizat aici faptul că, deşi în cazul calculatoarelor numerice în logică programată se vorbeşte de o structură fizică generală, aceasta nu poate fi total independentă de aplicaţia (sau familia de aplicaţii) pentru care este folosit calculatorul respectiv.

Deşi varianta de sistem de calcul în logică programată prezintă o serie de avantaje ce au dus la dezvoltarea rapidă şi extinderea aplicării acesteia în cele mai variate domenii, calculatoarele analogice rămân importante şi cu aplicaţii extinse.

Revenind la structura unui sistem de calcul în logică programată, existenţa programului indică faptul că trebuie să existe un element care să poată înţelege şi executa comenzile existente în acest program. Acest element se numeşte *unitate centrală* şi pe lângă funcţiile amintite va mai avea şi alte sarcini în sistemul de calcul.

Din cele prezentate s-a conturat o structură generală a unui sistem de calcul numeric în logică programată şi care trebuie să conţină: unitatea centrală, memoria şi interfeţele.

Intern, un calculator prelucrează mai multe fluxuri de informaţie dintre care principalele fluxuri sunt reprezentate de datele numerice şi de instrucţiunile programului. După modul de prelucrare al acestor fluxuri informaţionale calculatoarele se pot clasifica în:

- maşini de tip SISD (Single Instruction Single Data) care prelucrează la un moment dat o singură instrucţiune program şi o singură valoare numerică;
- maşini de tip SIMD (Single Instruction Multiple Data) care prelucrează la un moment dat o singură instrucţiune program dar mai multe fluzuri de date numerice;

- maşini de tip MIMD (Multiple Instruction Multiple Data) care prelucrează la un moment dat mai multe instrucțiuni program şi mai multe date numerice.

Maşinile de tip SIMD sau MIMD fac parte din categoria calculatoarelor paralele care pot prelucra în paralel mai multe fluxuri de informație.

Din punct de vedere al puterii de calcul, în prezent calculatoarele se clasifică astfel:

- microsisteme, sisteme simple bazate pe un microcontroler (calculator integrat), un procesor de semnal – DSP – (microcontroler destinat prelucrării digitale a semnalelor) sau un microprocesor (unitate centrală integrată pe un singur chip) destinate automatizărilor, dispozitivelor periferice, bunurilor de lar consum, comunicațiilor etc;
- calculatoare personale (PC – Personal Computer), pentru un singur utilizator, construite pe baza unui microprocesor. Aceste maşini sunt de cele mai multe ori de tipul SISD sau SIMD şi sunt prevăzute cu dispozitive de intrare (de exemplu: tastatură, mouse, joystick, scanner, etc.) pentru introducerea datelor de intrare, dispozitive de ieşire (de exemplu: display, imprimantă, plotter, etc.) pentru prezentarea datelor de ieşire şi dispozitive de stocare a datelor de intrare şi de ieşire (discuri, bandă magnetică, disc optic, etc.);
- stații de lucru (WS - WorkStation), pentru un singur utilizator, similare calculatoarelor personale dar dotate cu microprocesoare mai puternice şi cu monitoare de calitate fiind destinate prelucrărilor complexe;
- minicalculatoarele, destinate mai multor utilizatori suportând de la 10 la câteva sute de utilizatori simultan. Sunt maşini de tip MIMD;
- calculatoare de tip mainframe, calculatoare mai puternice decât minicalculatoarele, destinate utilizării simultane de către mai mulți utilizatori (multi user) putând suporta de la câteva sute la câteva mii de utilizatori;
- supercalculatoarele, calculatoare extrem de rapide care pot executa sute de milioane de instrucțiuni pe secundă.

Având în vedere importanța unei astfel de sistem de calcul cât şi perspectivele de dezvoltare şi aplicare, în continuare vom discuta principalele probleme legate de arhitectura acestora.

Sistemele de calcul în logică programată, larg răspândite în prezent, folosesc două tipuri de arhitecturi: arhitectura *von Neumann* şi arhitectura *Harvard*.

Aceste arhitecturi se numesc *arhitecturi secvențiale*, care au o singură unitate centrală şi care execută în mod secvențial programul aflat în memorie spre deosebire de *arhitecturile paralele* care sunt structuri realizate cu două sau mai multe unități centrale.

3.2. Arhitectura von Neumann

Cea mai simplă structură (structura minimală) a unui sistem de calcul este prezentată în figura 3.1.

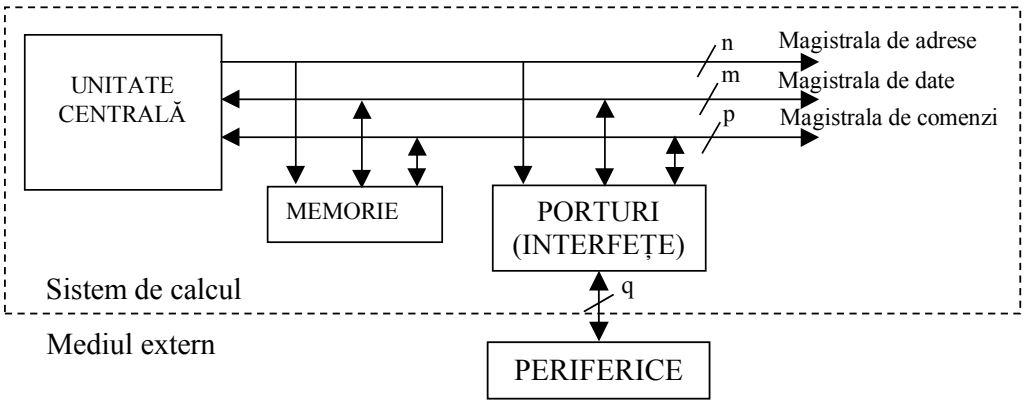


Fig. 3.1. Structura generală a unui sistem de calcul

Structura prezentată este structura minimală a unui sistem de calcul numeric numită şi arhitectură von Neumann. Evident, structura unui sistem de calcul poate conține şi alte elemente care vor duce la ridicarea performanțelor sistemului de calcul dar elementele prezentate în structura de mai sus sunt *indispensabile* funcționării unui sistem numeric în logică programată.

Trebuie observat faptul că această structură nu este legată de o anume aplicație ci ea este determinată doar de modul de funcționare al unui sistem numeric în logică programată (maşină von Neumann în acest caz) si va acoperi o arie largă de aplicații.

Elementele componente ale acestei structuri vor fi prezentate în continuare.

Unitatea centrală care are rolul de comandă şi control a sistemului de calcul şi de execuție a programelor ce se găsesc în memorie.

Un calculator poate avea o singură unitate centrală (in cele mai multe din cazuri) sau mai multe unități centrale ce lucrează în paralel. Sistemele de calcul cu o singură unitate centrală se numesc ***monoprocesor*** iar activitatea de prelucrare se numeşte ***monoprocesare*** iar cele care au mai multe unități de calcul ce lucrează în paralel se numesc ***multiprocesor*** iar activitatea de prelucrare a informației se numeste ***multiprocesare***. Din cauză că în continuare ne vom referi numai la calculatoarele monoprocesor nu se va mai specifica acest lucru explicit.

Memoria care are rolul de a stoca (păstra) programe şi date.

Porturile sau interfețele care au rolul de a realiza schimbul de informație dintre sistemul de calcul şi mediul extern prin trensformarea şi adaptarea semnalelor conform cerințelor.

Toate aceste elemente, care sunt realizate cu ajurorul unor circuite integrate pe scară largă şi foarte largă, sunt legate între ele prin intermediul unor conexiuni electrice. Prin aceste conexiuni circulă semnale electrice care au o anumită semnificație din punct de vedere al informației codificate (binar). Din acest motiv, conexiunile electrice sunt grupate după semnificația informației vehiculate şi un astfel de grup poartă numele de

magistrală (bus). Avem trei tipuri principale de magistrale: magistrala de adrese, magistrala de date şi magistrala de comenzi. Fiecare din aceste magistrale este alcătuită, aşa cum s-a arătat, din mai multe conexiuni electrice. În figura noastră, magistrala de adrese are ***n*** linii, magistrala de date are ***m*** linii iar magistrala de comenzi are ***p*** linii.

Perifericele nu aparţin structurii sistemului de calcul dar sunt menţionate aici deoarece sunt indispensabile funcţionării unui calculator. Vorbind la modul general, un periferic realizează conversia unei anumite forme de energie în energie electrică. Spre exemplu, tastatura, care este un periferic transformă energia mecanică cu care apăsăm tastele într-un semnal electric care este preluat de către interfaţă şi adaptat formatului intern propriu sistemului de calcul, iar monitorul (display-ul) transformă energia electrică în energie luminoasă.

Se observă că interfeţele sunt legate la rândul lor la periferice prin intermediul unor magistrale (de dimensiune ***q*** în figura noastră). Aceste magistrale sunt de diferite tipuri, au diferite dimensiuni şi poartă diferite denumiri legate de tipul interfeţei la care sunt legate.

Elementele care alcătuiesc structura sistemului de calcul se împart în două categorii: *elemente de comandă* (master) şi *elemente comandate* (slave). Într-o structură de calcul vom recunoaşte uşor elementele de comandă prin faptul că acestea pot genera adrese (sensul săgeţii magistralei de adrese este dinspre modulele de comandă spre cele comandate).

3.2. Arhitectura Harvard

Structura simplificată a arhitecturii Harvard este prezentată în figura 3.2.

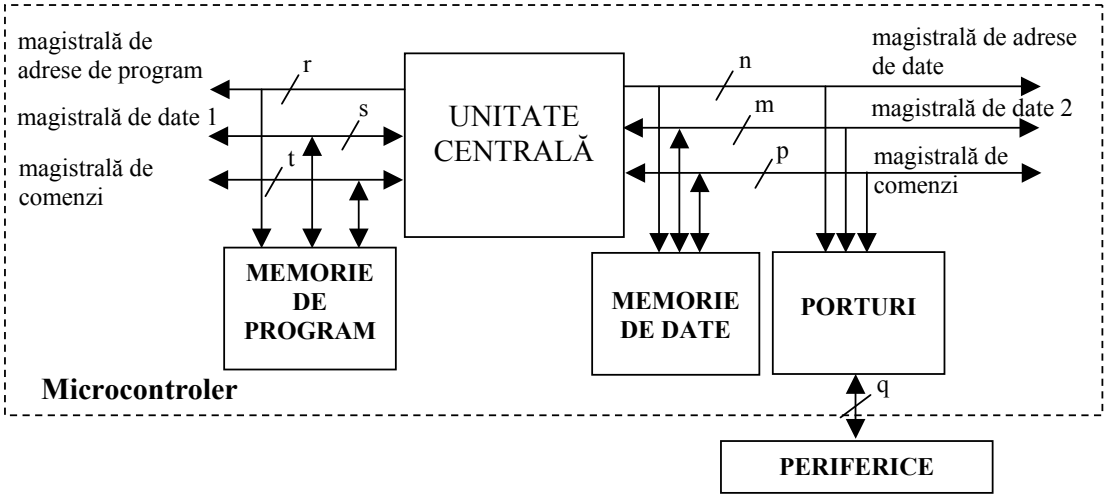


Figura 3.2. Arhitectura Harvard.

Principala caracteristică a acestui tip de arhitectură este reprezentată de utilizarea în structura sistemului de calcul a două memorii cu destinaţii diferite: o memorie de program şi o memorie de date. Acest lucru permite suprapunerea ciclului de extragere a codului operaţie din memorie cu ciclul de citire/scriere în memorie ceea ce duce la

creşterea corespunzătoare a vitezei de lucru. De asemenea astfel de sisteme folosesc cuvinte de lungimi diferite pentru codul instrucțiunii şi pentru date (de exemplu, coduri de instrucțiuni pe 14 biți şi date pe 8 biți) ceea ce permite utilizarea unui set complex de instrucțiuni.

Utilizarea memoriilor cu destinații diferite presupune existența unei magistrale de adrese de program şi a unei magistrale de adrese de date şi, de asemenea, existența unei magistrale de date pentru program – pe care sunt aduse codurile de instrucțiune – şi a unei magistrale de date pentru datele programului (fiecare dintre aceste magistrale putând avea dimensiune diferită).

O astfel de arhitectură este utilizată cu precădere în structura procesoarelor de semnal (DSP – Digital Signal Processor) sau a microcontrolerelor de tip Microchip PIC, Atmel AVR etc.

În astfel de sisteme, programul şi datele acestuia sunt stocate în memorii diferite. De regulă memoria program este o memorie de tip *FLASH* în aşa fel încât programul să nu se şteargă la întreruperea tensiunii de alimentare iar datele sunt stocate în memorii de tip *RAM* static şi memorii de tip *EEPROM*.

Asfel de sisteme reprezintă microcalculatoare integrate care au unitatea centrală, memoria şi porturile realizate pe un singur circuit integrat.

Un calculator poate avea mai multe magistrale de acelaşi tip care se deosebesc prin viteza de variație (frecvența) a semnalelor care parcurg aceste magistrale, dimensiunea şi destinația acestora. Din cauză că diferitele elemente conectate la magistrală au viteze de lucru diferite, elementele mai lente (cu viteză de lucru mai scăzută), vor impune frecvența maximă pe magistrale. Pentru creşterea performanțelor unui calculator, magistralele se realizează pe mai multe nivele cu frecvențe de lucru diferite. Astfel, pe magistrala cu viteza cea mai mare se conectează de obicei unitatea centrală şi memoriile rapide iar pe magistralele cu viteză mai scăzută se conectează memoriile lente şi porturile. Comunicația între magistralele ce lucrează la frecvențe diferite se realizează cu ajutorul unor circuite integrate specializate numite *controlere de magistrală*. Rezultă că un criteriu de performanță a unui calculator este reprezentat de frecvența maximă a magistralelor interne.

Diferitele structuri particulare de calculatoare pot conține şi alte elemente în afară de cele prezentate în figura 3.1, în scopul creşterii performanțelor calculatorului sau datorită destinației acestuia. Dintre aceste elemente suplimentare cele mai uzuale sunt: circuitul de acces direct la memorie (DMA), controlerul de întreruperi, controlerul video şi controlerul de comunicație. Este de remarcat aici faptul că în cazul calculatoarelor de proces apar în plus ceasul de timp real şi ceasul de gardă (watchdog timer) necesare rulării în timp real a aplicațiilor. În figura 3.3 este prezentată schema bloc a unui calculator de proces. În această figură se vede că sistemul este organizat în jurul a mai multor magistrale. Pe magistrala A se află sistemul de calcul propriu-zis iar pe magistrala B se află circuitele de interfață cu procesul. Fiecare dintre magistrale sunt alcătuite, la rândul lor, din mai multe magistrale care nu sunt arătate în figurră.

Pentru a evita confuziile, trebuie menționat faptul că în figura 3.3, printr-un abuz de limbaj, unitate centrală este denumită placa electronică, notată cu 880-P, ce conține atât unitatea centrală propriu-zisă cât şi porturi de intrare/ieşire. Tot aşa, prin extinderea acestei expresii, unitate centrală este denumită cutia sau dulapul în care se află această componentă din structura sistemului de calcul.

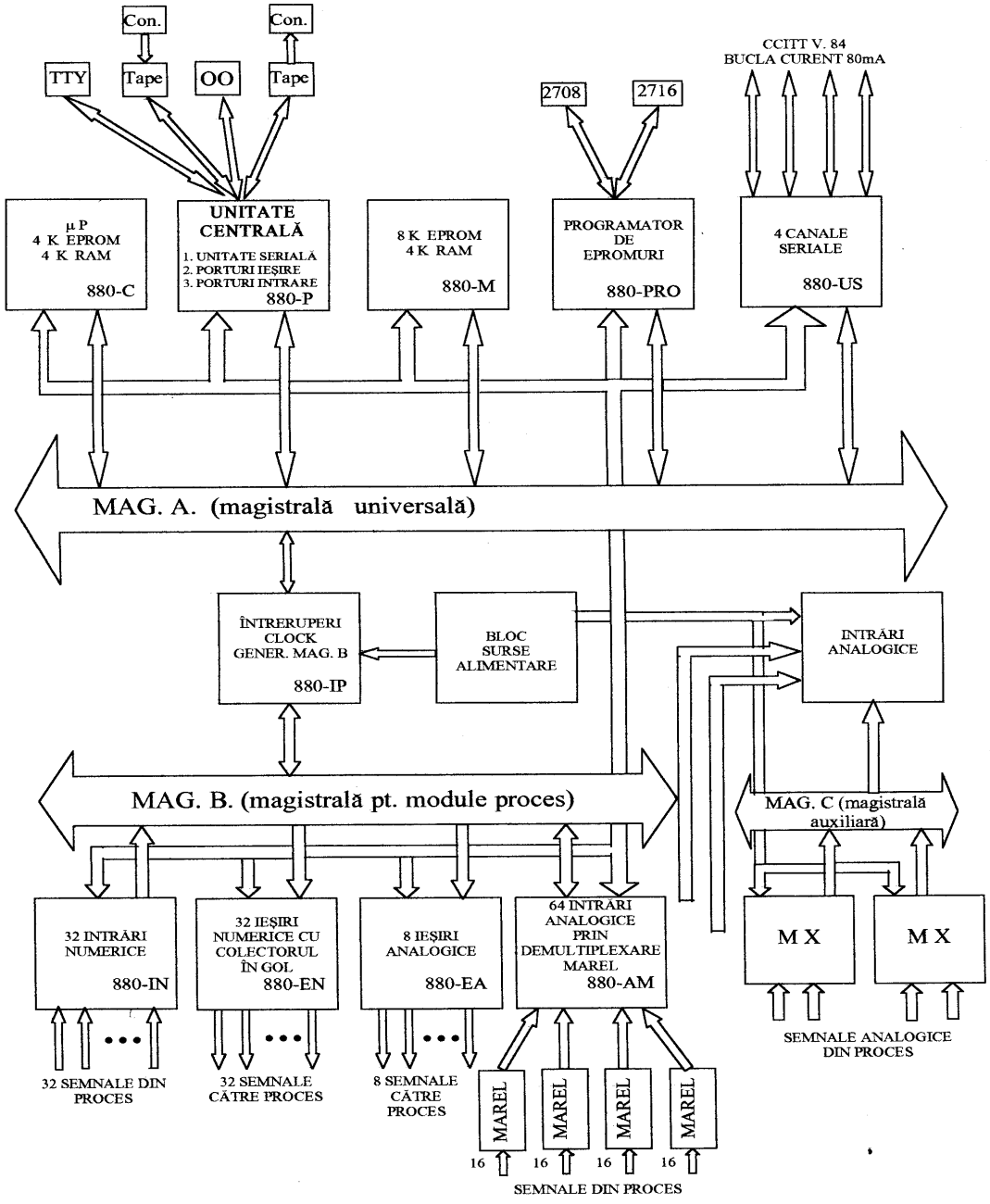


Fig. 3.3. Structura unui calculator de proces

CAPITOLUL 4

UNITATEA CENTRALĂ

4.1. Introducere

Unitatea centrală reprezintă componenta principală a sistemului care coordonează toate activitățile acestuia. De performanțele unității centrale depind în mod esențial performanțele calculatorului. La calculatoarele personale unitatea centrală este reprezentată de un singur circuit integrat numit microprocesor. Stațiile de lucru conțin unul, două sau mai multe microprocesoare, conectate într-o arhitectură scalară, care lucrează în paralel. În cazul minicalculatoarelor, a calculatoarelor mainframe sau a supercalculatoarelor, unitatea centrală este realizată cu ajutorul mai multor componente. De obicei sunt utilizate integrate de tip bit-slice care reprezintă o parte (felie) dintr-o unitate centrală. Prin conectarea în paralel a mai multor astfel de circuite se obțin unități centrale cu diferite performanțe. Aceste unități sunt microprogramabile de către utilizator ceea ce le conferă flexibilitate în exploatare dar complică procesul de proiectare. Microprogramarea este o metodă de realizare a automatelor de comandă prin înscrierea programelor acestora într-o memorie. Prin microprogramare se obțin microprograme, constituite din microinstrucțiuni. Microprogramarea presupune o cunoaştere de detaliu a echipamentului căruia îi sunt destinate microprogramele. Spre deosebire de marea majoritate a altor metode de programare instrucțiunile utilizate în microprogramare conțin atât codul operației cât şi eventualii operatori, cât şi adresa instrucțiunii următoare.

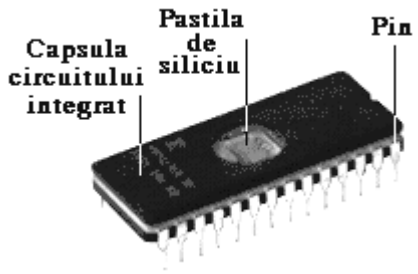


Fig. 4.1. Circuit integrat

Fig. 4.2. Microprocesorul INTEL 486

În figura 4.1 este prezentat modul de realizare a unui circuit integrat.

Principalele mărimi caracteristice ale unei unități centrale sunt reprezentate de: setul de instrucțiuni, numărul de biți prelucrați simultan (lărgimea magistralei de date) şi frecvența de ceas (viteza la care funcționează unitatea centrală).

Din punct de vedere al setului de instrucțiuni, unitățile centrale sunt de două categorii: unități centrale de tip RISC (Reduced Instruction Set Comuter) cu un număr

relativ redus de instrucțiuni şi unități centrale de tip CISC (Complex Instruction Set Computer) cu un număr mare de tipuri de instrucțiuni.

Unitățile de tip RISC sunt ieftine şi foarte rapide din cauză că setul simplu de instrucțiuni le permite execuția cu viteză mare a programului. Unitățile centrale de tip CISC au avantajul posibilităților extinse de lucru datorită setului complex de instrucțiuni iar în ultimul timp, ele sunt un concurent serios, la viteză, al unităților centrale de tip RISC. Cunoscutul microprocesor al firmei INTEL, Pentium, este considerat ca fiind o unitate centrală de tip CISC deşi conține în arhitectura sa multe elemente ale unei maşini RISC. În figura 2.4 este prezentat microprocesorul INTEL 486.

Numărul de biți prelucrați simultan de unitatea centrală reprezintă de fapt lățimea magistralei de date din cauză că pe această magistrală sunt vehiculate date şi instrucțiuni. Lățimea magistralei de date este de regulă un multiplu de opt, în prezent fiind întâlnite în mod uzual unități centrale ce lucrează pe 8, 16, 34, 64, 128 sau 256 de biți.

Frecvența semnalului de tact (semnalul de ceas) aplicat unității centrale reprezintă de asemenea un criteriu de performanță a acesteia. Având în vedere faptul că o instrucțiune este executată de unitatea centrală în una sau mai multe perioade de ceas, rezultă că viteza de calcul creşte odată cu creşterea frecvenței de ceas. În prezent calculatoarele personale performante utilizează frecvențe de tact de până la 600MHz cu perspective de a ajunge la 1GHz.

Aceste criterii de performanță nu sunt absolute din cauză că ele depind în mare măsură de arhitectura unității centrale. Astfel, spre exemplu, două unități centrale cu aceeaşi frecvență de ceas, pot merge cu viteze diferite, datorită structurii interne, cu până la de 20 de ori.

Un alt criteriu de performanță îl reprezintă dimensiunea memoriei interne, memoria cache, a unității centrale. Memoria cache este o memorie foarte rapidă care este folosită ca intermediar între unitatea centrală şi memoria principală a sistemului. Această memorie poate fi atât internă (în structura unității centrale) cât şi externă. Din acest motiv se spune că memoria cache este organizată pe nivele. Cu cât dimensiunea memoriei cache interne este mai mare, cu atât performanțele unității centrale vor fi mai bune.

Alte criterii de performanță ce pot fi luate în considerare în cazul unităților centrale sunt: numărul maxim adresabil de porturi de intrare/iesire, numărul şi dimensiunea registrelor interne, şi modalitățile de adresare.

O unitate centrală execută o instrucțiune în unul sau mai mulți ciclii maşină. Un ciclu maşină reprezintă o activitate elementară a unității centrale (extragerea din memorie a codului operației, citire/scriere din/în memorie, citire/scriere din/în port, achitare cerere întrerupere etc. Un ciclu maşină poate dura una sau mai multe perioade a ceasului de comandă al unității centrale.

Pentru determinarea performanțelor unităților centrale se foloseşte o metodă bazată pe detrminarea numărului de operații în virgulă mobilă (floating-point operatins) executate de aceasta într-o secundă. Unitatea de măsură este FLOPS (FLoating-point Operations per Second), în mod obişnuit performanțele unităților centrale fiind măsurate în megaFLOPS sau gigaFLOPS.

La punerea sub tensiune unitatea centrală porneşte dintr-o stare inițială cunoscută începând să execute programul aflat în memorie de la o adresă bine precizată. Pe timpul execuției unui program unitatea centrală poate ajunge dintr-un motiv sau altul într-o stare nedorită care să afecteze modul corect de funcționare al calculatorului. În aceste situații unitatea centrală trebuie readusă în starea inițială. Acest lucru se face fie în mod automat de către circuitele specializate ale calculatorului fie de către utilizator prin apăsarea unui buton (RESET). Este evident faptul că inițializarea unității centrale poate fi făcută şi prin întreruperea tensiunii de alimentare dar acest lucru este nerecomandabil de cele mai multe ori pentru că se pot pierde informații în curs de prelucrare de către periferice şi/sau informații legate de prelucrările curente. De asemenea şocurile termice şi de tensiune ce apar la pornirea calculatorului nu recomandă folosirea acestui procedeu pentru reinițializarea unității centrale. Butonul de inițializare a unității centrale (RESET) este marcat uneori şi cu simbolul: ▸●◂.

Datorită dezvoltării spectaculoase a microsistemelor şi datorită faptului că unitățile centrale a acestora, microprocesoarele, beneficiază de o întreagă gamă de inovații tehnologice, ne vom concentra atentia asupra unitaților centrale de tip microprocesor

Principalele avantaje oferite de microsisteme față de sistemele convenționale sunt:

- costul scăzut al hardware-ului;
- siguranța în funcționare îmbunătățită;
- interferențe electromagnetice scăzute - nivelul larg de integrare evită influiențele electromagnetice chiar şi la nivele mari de variație ale tensiunii şi curentului prin circuitele electronice de putere. În general pentru protecție este suficient ecranul cu care prevăzută componenta. Zgomotul de cuplare cu sursele de alimentare şi semnalele de intrare poate fi minimizat printr-o bună filtrare analog-digitală;
- absența împrăştierii sau a variației parametrilor;
- compatibilitate cu controlul digital ierarhic;
- hardware şi software universal;
- posibilitatea de diagnostic şi autodiagnostic.

Principalele dezavantaje prezentate de către microsisteme sunt:

- viteza de răspuns (calcul) este în unele cazuri insuficientă pentru elaborarea unui răspuns în timpul aşteptat;
- eroarea de cuantizare - eroarea de cuantizare poate fi micşorată prin creşterea numărului de biți ai cuvintelor microsistemului şi ai convertorului A/D;
- lipsa de acces la semnale software - nu se pot efectua măsurători cu osciloscopul asupra semnalelor software şi deci trebuie concepute măsuri adecvate de depanare.
- dezvoltarea software-ului poate fi uneori scumpă.

În cazul microsistemului cu microprocesor unitatea centrală şi celelalte elemente ale microsistemului sunt componente distincte fapt ce participă la flexibilitatea soluției. În

cazul microcontrolerelor într-o singură componentă sunt integrate pe lângă unitatea centrală şi o parte din elementele microsistemului. Deşi microcontrolerele prezintă o flexibilitate mai redusă ele prezintă avantajul miniaturizării şi a preţului de cost scăzut.

4.2. Microprocesorul universal (structura generală a unui microprocesor)

Structura unui microprocesor cuprinde: o unitate aritmetică şi logică, având drept scop efectuarea de operaţii aritmetice şi logice elementare; un set de registre pentru memorarea temporară şi manipularea cu viteză ridicată a unui numar relativ mic de rezultate intermediare; un bloc de comandă şi secvenţiere care asigură desfăşurarea ordonată a tuturor operaţiilor în interiorul microprocesorului, precum şi comunicaţia acestuia cu lumea exterioară; un bloc de decodificare a instrucţiunii curente, care interpretează instrucţiunea în curs şi determină acţiunile ce se impun; un bloc de tratare a cererilor de întrerupere uilizat pentru luarea în considerare a evenimentelor asincrone faţă de desfăşurarea programului; tampoane între microprocesor şi magistralele sistemului. Microprocesoarele din ultimile generaţii au inclus şi un coprocesor matematic ce le permite realizarea unor calcule matematice complexe cu viteză ridicată şi memoria cache.

Structura de principiu a microprocesorului universal este prezentată în figura 4.3.

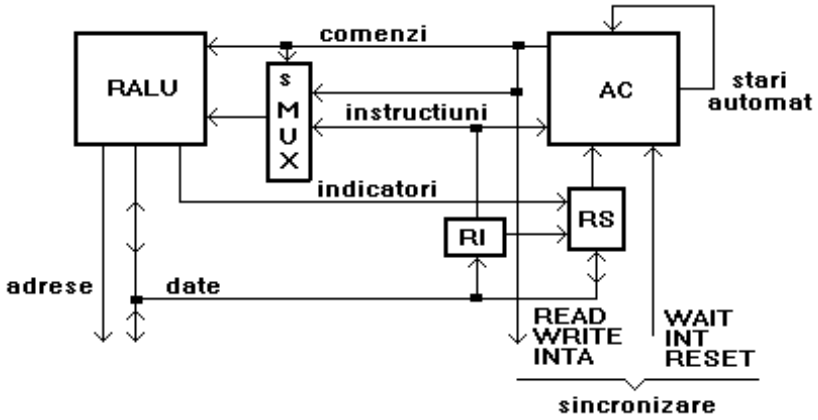


Fig.4.3. Schema bloc a microprocesorului universal

Această structură este formată din următoarele elemente componente:

- RALU - unitatea logică şi aritmetică şi regiştrii de uz general şi special;
- AC - automatul de control;
- RI - registrul de instrucţiune;
- RS - registrul de stare;
- MUX - un grup de multiplexoare ce asigură o parte din comenzile la bornele RALU de la AC sau RI, sub controlul (selecţia) semnalelor din AC.
- Semnalele interne care asigură funcţionarea sistemului sunt următoarele:
  - comenzi, generate de AC către subansambluri sau către exterior;

- instrucțiuni, generate din RI către AC şi prin intermediul MUX, către RALU;
- indicatori, generați de ALU din RALU către RS;
  - date - reprezintă un bus intern bidirecțional pe care sunt cuplate RALU, RS şi RI (unidirecțional);
  - adrese - cale de adrese ce se emite în exterior din RALU pentru controlul dispozitivelor de memorare exteroare microprocesorului.
  - Semnalele la bornele microprocesorului sunt următoarele:
  - adrese - cale unidirecțională;
  - date - cale bidirecțională;
  - semnale de sincronizare:
    - READ - semnal ce comandă citirea de la adresă configurației binare de pe calea date;
    - WRITE - semnal ce comandă la adresă a configurației binare de pe calea date;
    - WAIT - semnal recepționat ce impune trecerea microprocesorului într-o stare de aşteptare drept urmare a unei comenzi READ sau WRITE ce nu a fost încă executată;
    - INT - semnal de întrerupere a cărui recepționare stopează procesul curent de calcul pentru a da curs unui eveniment exterior sistemului;
    - INTA - semnal ce indică luarea în considerație a semnalului de întrerupere INT; ca urmare a acestui semnal, dispozitivul ce a activat INT va genera pe date o configurație binară specifică;
    - RESET - semnal de inițializare a funcționării microprocesorului.

Microprocesoarele sunt procesoare la care funcția de control intern este minimizată, în sensul că automatul de control AC este gândit foarte simplu. Procesoarele microprogramate se situează la extrema în care funcția de control intern are o pondere foarte importantă. Dacă un microprocesor foloseşte cât mai direct codul instrucțiunii (biții din RI comandă direct funcționarea RALU), în cazul imensei majorități a instrucțiunilor, la o structură microprogramată, codul instrucțiunii acționează cu preponderență asupra automatului de control, declanşând secvențe complexe de comandă.

4.3. Caracteristicile principalelor tipuri de microprocesoare

În continuare sunt prezentate pe scurt principalele caracteristici ale unor tipuri reprezentative de microprocesoare.

4.3.1. Microprocesorul ZILOG Z80

Deşi este un microprocesor care a apărut de o perioadă de timp destul de lungă el este unul dintre microprocesoarele pe 8 biți dintre cele mai populare. Pentru acest tip de microprocesor au fost dezvoltate o serie foarte mare de aplicații începând de la bunuri de larg consum până la aplicații industriale.

Microprocesorul Z80 împreună cu componentele sale auxiliare formează o familie tipică pentru generaţia de microprocesoare de 8 biţi. Componentele din familie asigură posibilitatea realizării tuturor funcţiilor clasice ale unui sistem cu microprocesor.

Principalele caracteristici ale acestui microprocesor sunt:

- magistrala de date pe 8 biţi;
- magistrala de adrese pe 16 biţi;
- 12 regiştrii de uz general pe 8 biţi cu utilizare pe 16 biţi, regiştrii cu selectare alternantă câte şase;
- 2 regiştrii acumulator şi 2 regiştrii indicatori de condiţii cu selectare alternantă;
- 2 regiştrii speciali, unul pentru determinarea vectorului de întrerupere şi unul pentru reîmprospătarea transparentă a memoriei dinamice;
- 2 regiştrii index pe 16 biţi;
- o întrerupere nemascabilă şi una mascabilă cu un mecanism foarte flexibil de răspuns la cererea de întrerupere;
- este capabil să execute operaţii aritmetice (mai puţin înmulţirea şi împărţirea) şi logice;
- frecvenţă de tact maximă 6MHz.

4.3.2. Microprocesoarele INTEL 80x86

Familia microprocesoarelor INTEL s-a îmbogăţit continuu cu noi membrii care au dus la creşterea performanţelor hardware şi software a acestor componente. Principalul avantaj al acestei familii de microprocesoare este păstrarea compatibilităţii software între diferitele generaţii începând cu microprocesorul 8086 ceea ce a permis dezvoltarea continuă a aplicaţiilor pentru aceste microprocesoare.

Începând cu microprocesorul 80286 s-au introdus două moduri de funcţionare diferite: ***modul real*** (Real Address Mode) în care microprocesorul poate adresa ca şi 80806 un spaţiu de memorie de 1Mo şi ***modul protejat*** (Protected Virtual Address Mode) în care spaţiul adreselor fizice a crescut la 16Mo iar cel al adreselor virtuale la 1Go. Microprocesorul 80286 a fost proiectat pentru a permite (în mod protejat) ***sisteme multitasking***, oferind câte un segment de stare ataşat fiecărui ***task***, care sunt structuri manevrate hardware şi conţin stările curente (incluzând toţi regiştrii) ale taskurilor. Selectoarele acestor segmente de stare identifică unic taskul ataşat. De asemenea, este facilitată şi comutarea taskurilor, care poate fi invocată printr-o singură instrucţiune. Fiecare task din sistem poate avea propriul lui spaţiu de adrese logice. Microprocesorul 80286 oferă şi un mecanism evoluat pentru comunicarea între taskuri, sincronizarea lor şi partajarea memoriei.

Istoria microprocesoarelor din familia 80x86 este strâns legată de cea a coprocesoarelor matematice. Fiecărui element de bază din familie i-a fost asociat un coprocesor pentru a îmbunătăţi performanţele sistemului de calcul. Pentru microprocesoarele 8086, 80286 şi 80386 sunt folosite coprocesoarele 8087, 80287 şi respectiv 80387.

Microprocesorul 80386 pe 32 de biţi a fost proiectat pentru utilizarea în aplicaţii intensive complexe. Datorită celor 32 de biţi de adrese, spaţiul adreselor fizice este de 4Go iar spaţiul adreselor logice de 64To. Este prevăzut cu microcod care suportă direct aplicaţii care utilizează întregi mari, structuri complexe de date şi un număr mare de programe. Microprocesorul are patru moduri distincte de funcţionare: real, virtual 8086, protejat 286 şi nativ 386. În figura 4.4 este prezentată schema bloc a microprocesorului 80386.

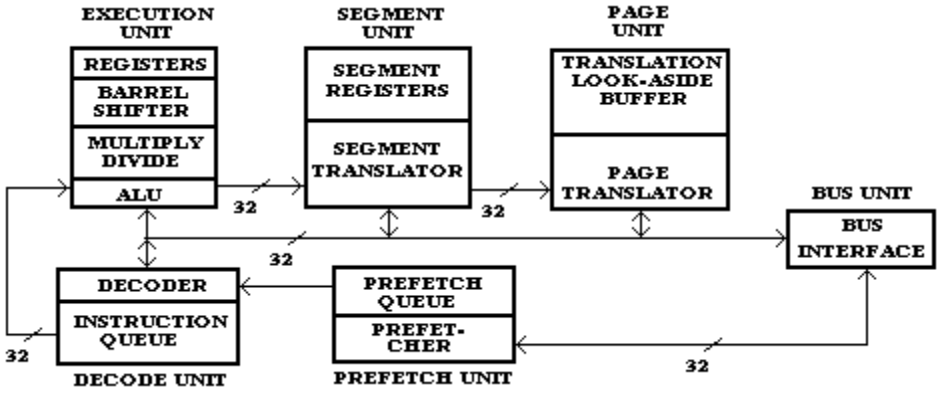


Fig. 4.4. Schema bloc a microprocesorului 80386

Microprocesorul 80486 oferă multe facilităţi noi: o memorie cache de 8ko, o unitate de gestiune a memoriei compatibilă 80386, un procesor 80386 şi un subset 80387 pe un singur chip, ceea ce permite ca softul existent (fiind vorba de înalta compatibilitate) să ruleze mai repede pe 80486 decât pe perechea 80386-80387.

Microprocesoarele de tip INTEL, începând cu microprocesorul de tip 80386 au patru moduri de funcţionare de bază: modul real, modul virtual şi modul protejat. Având în vedere faptul că sistemul de operare DOS este destinat în exclusivitate microprocesoarelor de tip INTEL şi datorită faptului că aceste mecanisme se întâlnesc şi la alte tipuri de microprocesoare, ele vor fi descrise pe scurt în continuare.

MODUL REAL. Acest mod este cel în care se intră după iniţializarea microprocesorului. De regulă, sub sistemul de operare DOS microprocesorul se află în modul real. Specific acestui mod de funcţionare este faptul că microprocesorul nu poate rula decât un singur program odată. Există posibilitatea rulării programelor sub sistemul de operare DOS şi în alte moduri ale microprocesorului, dar acest lucru necesită extensii ale sistemului de operare (existând în acest sens programe specifice).

MODUL VIRTUAL. Acest mod permite execuţia programelor în contextul mecanismelor de protecţie, gestiune a taskurilor şi management al memoriei. Un program poate rula în mod virtual în paralel cu alte programe care rulează în mod protejat. De asemenea, pot fi executate deodată mai multe programe în mod virtual. Sistemul de operare Windows 9x se bazează pe acest mod performant de funcţionare al microprocesorului, permiţând execuţia în paralel a mai multor programe (sesiuni DOS) cât şi a mai multor programe Windows.

MODUL PROTEJAT. În acest mod, microprocesorul emulează funcționarea unui microprocesor de tip 80286 în mod protejat. Modul protejat prezintă un mecanism sofisticat pentru protejarea datelor, integritatea sistemului, concurența taskurilor şi gestiunea memoriei incluzând şi cea a memoriei virtuale. În modul protejat, în cazul microprocesorului 80286, spațiul adreselor fizice este mărit de la 1 Moctet la 16 Mocteți, în timp ce spațiul adreselor virtuale a fost mărit la 1 Goctet. În acest mod, programele folosesc adrese virtuale, translatarea lor în adrese fizice făcându-se automat pe baza unor tabele cu descriptori de segmente. Acest mecanism permite implementarea eficientă a sistemelor cu memorie virtuală în care utilizatorul vede memoria internă şi cea externă ca o singură memorie. Modul protejat permite implementarea sistemelor multitasking, oferind câte un segment de stare ataşat fiecărui task, care sunt structuri manevrate hardware şi conțin stările curente (incluzând toți regiştrii) ale taskurilor. Selectoarele acestor segmente de stare identifică unic taskul ataşat. De asemenea, este facilitată şi comutarea taskurilor, care poate fi invocată printr-o singură instrucțiune. Fiecare task din sistem poate avea propriul lui spațiu de adrese logice, existând de asemenea un mecanism evoluat pentru comunicarea între taskuri, sincronizarea lor, partajarea memoriei etc.

MODUL NATIV. Acest mod foloseşte întreaga putere a microprocesorului. Memoria virtuală permite ca dimensiunea unui program să fie limitată de spațiul pe disc şi nu de dimensiunea memoriei interne. Mecanismele de protecție sunt destul de puternice pentru a evita accidentele între taskuri sau între utilizatori. Spațiul de adrese al taskurilor este complet separat, segmentele sunt tipizate şi au diferite drepturi de acces. De asemenea, se verifică şi depăşirea limitelor segmentelor. Există trei nivele de prioritate, sistemul de operare având nivelul zero (cel mai prioritar), iar aplicațiile programatorilor având nivelul trei (cel mai puțin prioritar).

Microprocesorul INTEL Pentium

Microprocesorul Pentium este integral compatibil cu procesoarele INTEL anterioare, dar se deosebeşte de acestea în multe privințe. Cel puțin una din aceste deosebiri este majoră: microprocesorul Pentium are două canale identice de procesare a datelor, ceea ce îi permite să execute două instrucțiuni în acelați timp. Această capacitate de a executa simultan două instrucțiuni este numită tehnologie superscalară (procesare paralelă). Această tehnologie asigură performanțe suplimentare față de cea a procesorului 486.

Prin modul numit BTB (Branch Target Buffer) care utilizează o tehnică numită branch prediction (predicția salturilor) în scopul reducerii timpului de aşteptare în canalele de procesare, cauzat de aducerea instrucțiunilor unei ramuri aflate la o altă locație de memorie, se realizează la microprocesorul Pentium o utilizare mai eficientă a memoriei cache interne. Modul BTB încearcă să prevadă când va apare o instrucțiune de salt şi să aducă în memorie instrucțiunile

corespunzătoare ramurii la care se va face saltul. Utilizarea tehnicii de prevedere a ramificării unui program permite microprocesorului să mențină în funcționare la viteză maximă, cele două canale pentru execuția instrucțiunilor.

Principalele caracteristici ale microprocesorului Pentium sunt:

- gama frecvențelor maxime: 450MHz;
- dimensiunea regiştrilor: 32 biți;
- magistrala de date externă: 64 biți;
- magistrala memoriei 32 biți;
- memoria maximă: 4Gb;
- memorie cache încorporată: 8kb pentru instrucțiuni şi 8kb pentru date la primele tipuri ajungând la 2Mb în prezent;
- tipul memoriei cache încorporate: cu două blocuri asociate, Write-Back Data;
- coprocesor matematic cu unitate de calcul în virgulă mobilă (FPU) inclusă;
- gestionarea memoriei: SMM (System Management Mode), extinsă la a doua generație.

Dezvoltarea microprocesoarelor a depaşit de mult acest prag. În prezent există microprocesoare cu mai multe unități centrale integrate care au mecanisme sofisticate.

Datorită faptului că înțelegerea funcționării unui microprocesor este pe deplin posibilă prin analiza funcționării microprocesorului INTEL 8086, datorită structurii simple dar revoluționare a acestuia, compatibilitatea cu generațiile INTEL dezvoltate ulterior şi datorită existenței unei consistente documentații în legătură cu acest domeniu, vom prezenta în continuare structura în detaliu a acestui microprocesor.

4.3.2.1. Microprocesorul INTEL 8086/8088

Microprocesoarele 8086 şi 8088 de generația a III-a sunt microprocesoare cu o structură pe 16 biți, 8088 este proiectat cu o magistrală de date externă de 8 biți în timp ce 8086 poate transfera 16 biți deodată. Datorită faptului că în afară de această deosebire funcționarea celor două tipuri de microprocesoare este identică, în continuare ne vom referi numai la microprocesorul 8086. Performanțele acestor microprocesoare sunt datorate structurii interne de 16 biți şi arhitecturii ***pipeline*** care permite instrucțiunilor să fie preîncărcate în timpul ciclurilor disponibile ale magistralei.

Principalele caracteristici ale acestor microprocesoare sunt:

- magistrala de date de 16 biți;
- magistrala de adrese de 20 biți;
- adresare segmentată a memoriei;
- 4 regiştrii de uz general pe 16 biți adresabile direct sau pe octeți;
- 4 regiştrii de segment pe 16 biți;
- 2 regiştrii index pe 16 biți;
- un sistem foarte flexibil de adresare a memoriei (7 moduri);

- sistem de întreruperi hardware şi software;
- posibilitatea funcționării în sisteme multiprocesor;
- posibilitatea efectuării operațiilor aritmetice (inclusiv înmulțire şi împărțire) şi operații logice;
- frecvență de tact 6MHz.
-

Microprocesorul 8086 este un microprocesor pe 16 biți şi memorie adresabilă direct de 1Mo. Structura unității centrale, cu elementele accesibile programatorului, este prezentată în figura 4.5.

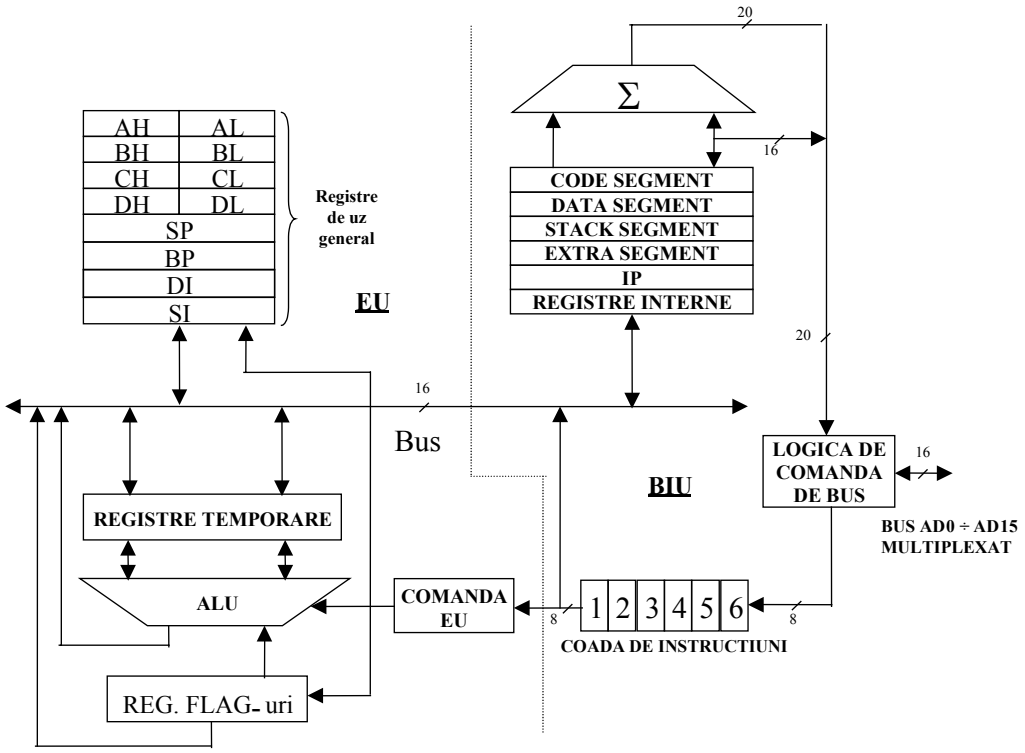


Figura 4.5. Structura unității centrale 8086.

Asa cum se arată în figura 4.5, structura microprocesorului I 8086 se compune din două unități : EU = unitate de execuție şi BIU = unitate de interfață cu magistrala (bus).

BIU execută toate ciclurile de bus (“READ”, “WRITE”, INTA”) fie la cererea EU sau pentru umplerea cu coduri a unei cozi Q de instrucțiuni. Coada de instrucțiuni este o memorie de tip FIFO cu 6 cuvinte.

BIU execută noi cicluri “FETCH” în intervalele cât EU nu solicită bus-ul.

EU obține coduri de la BIU (aşteaptă dacă Q este vidă), execută instrucțiunile, lucrând cu adrese şi date cu 8/16 biți, actualizează flag-urile şi furnizează adrese şi date către BIU.

EU calculează adresele efective ale operanzilor, conform modului de scheme utilizat. Adresa efectivă pe 16 biți, BIU generează adresa fizică cu 20 de biți, selectarea adresei efective, generate de EU.

Dacă Q este plină şi EU nu solicită transferuri pe bus apar cicluri în rotire (aşteptare).

Exemplu: T1 T2 T3 T4 TI TI ... TI TI T1 T2 T3 T4 ... TI = Idle State.

Modurile de lucru posibile pentru unitatea cuntrală I8086 sunt :

- Modul "minim" MN / MX = 1

Semnale de comandă: ALE, DT/K, DEN, M/IO, WR, INTA, HOLD, HOLDA.

Folosit în sisteme mici, de regulă monoprocesor. În figura alăturată circuitele "buffer"de tip 8286 pot lipsi dacă încărcarea electrică a bus-ului este redusă.

- Modul "maxim" MN / MX =0

UCP generează către controlerul de sistem I 8288 semnale de stare identificare ale ciclului de bus, în rest generează semnale corespunzătoare pe bus-ul de comandă.

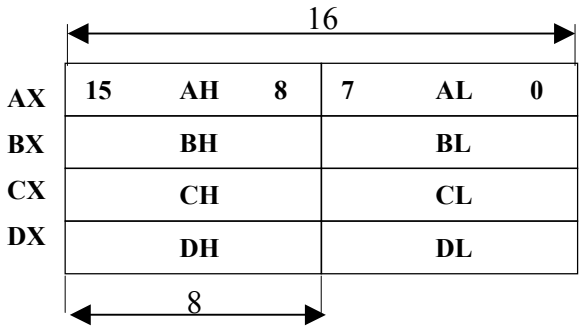
Conectarea UCP – I 8086 în modul "maxim".

În modul "maxim" liniile RQ/GT0, RQ/GT1, LOCK, QS0, QS1 se pot utiliza pentru conexiuni de tip multiprocesor. Se pot, de exemplu, conecta alte procesoare pe bus-ul local al UCP, utilizând pentru arbitrajul de bus liniile RQ/GT (care înlocuiesc pe HOLD/HOLDA din modul "minim").

Registrele unității centrale

Registrele unității centrale aflate în secțiunea **EU** (**E**xecution **U**nit) sunt:

AX, BX, CX, DX
SP, BP, SI, DI
F



Registre cu 16 biți de uz general, adresabile direct sau pe octet:

AX = (AH, AL)
BX = (BH, BL)
CX = (CH, CL)
DX = (DH, DL)

} fiecare din ele poate servi ca destinație a datelor (accumulator)

Utilizările implicite ale registrelor sunt:

AX: utilizat pentru operaţii aritmetice (*), (/) pe 16 biţi şi pentru operaţii de I/E pe 16 biţi; în mod analog AL este utilizat pe 8 biţi şi în plus pentru aritmetică zecimală şi conversii de cod; AH este utilizat la (*) şi (/) pe 8 biţi;
BX: utilizat în conversii de cod şi ca registru de bază de adrese;
CX: utilizat în operaţii cu şiruri, cu rol de contor de cicluri;
CL: utilizat în deplasări (stânga, dreapta – cu un număr de paşi daţi ca parametru de valoare lui CL);
DX: utilizat la (*), (/) pe 16 biţi şi ca registru de adresare indirectă la porţile de intrare – ieşire (I/E);
SP: utilizat implicit în toate operaţiile cu stiva;
SI, DI: utilizate în operaţiile asupra şirurilor de date;

F = registrul de flag-uri şi control al procesorului (se află în EU).
Semnificaţia fanioanelor din registrul **F** este:

CF = C, PF = P, AF = H, ZF = Z, SF = S – semnificaţii obişnuite :
CF = "carry flag": depăşire aritmetică;
PF = “parity flag”: paritate;
AF = “auxiliary flag”: transport între bitul 3 şi 4;
ZF = "zero flag”: valoare zero;
SF = “sign flag”: semnul.
TF = “trip flag”; TF = 1 determină UCP să lucreze în mod pas cu pas(“single step”), în care CA generează o întrerupere internă după fiecare execuţie a unei instrucţiuni;
IF = masca pentru întreruperi externe (IF =1 => validarea întreruperilor; IF =0 => invalidarea întreruperilor);
DF = “direction flag”- indică direcţia deplasării adresei la operaţiile cu şiruri de date (DF =1 => autodecrementare, DF =0 => autoincrementare, după o operaţie elementară);
OF = V (depăşire).

Registrele **SP**, **BP** sunt registre cu 16 biţi utilizate în operaţiile cu stiva.
Registrele **SI**, **DI** sunt registre cu 16 biţi utilizate în operaţiile cu şiruri; **SI** conţine adresa sursei iar **DI** adresa destinaţiei.

Registrele aflate în secţiunea **BIU** (**B**us **I**nterface **U**nit) a unităţii centrale 8086 sunt:

CS, DS, SS, ES = sunt registre segment care conţin adresele de bază ale segmentelor logice de cod, date, stivă şi extrasegment;
IP = Instruction Printer = contor de program, cu 16 biţi. Valoare ce reprezintă adresa relativă (offset-ul) a instrucţiunii curente în segmentul de cod (relativ la CS). În cazul unei instrucţiuni de salt, IP este salvat în vârful

stivei (împreună cu CS, deci saltul este inter-segment) şi apoi încărcat cu adresa relativă în segmentul de cod a instrucţiunii ţintă ;

Posibilităţile de lucru în sisteme multiprocesor

Arhitectura familiei I8086 conlucrarea între două tipuri de procesoare:

- independente (execută propriile secvenţe de instrucţiuni) ;
- coprocesoare – care obţin instrucţiuni din memoria unui procesor gazdă (host), urmărind ciclurile "FETCH"ale gazdei, le recunosc pe cele destinate lor şi le execută. Efectul obţinut este extinderea setului de instrucţiuni al gazdei.

Organizarea ierarhică a bus-urilor

Familia admite două tipuri de bus-uri: locale şi de sistem. Ambele pot fi multimaster (mai multe UCP conectate la bus). Între bus-ul local şi cel de sistem se conectează interfeţe.

Conexsiunile externe ale I8086

În modul "maxim" UCP livrează controlerului de sistem I 8288, în fiecare ciclu de bus, un cuvânt de comandă (S0, S1, S2), pe care acesta îl decodifică şi generează semnale de comandă pe magistrală (bus). În acest mod se pot conecta mai multe UCP la bus.

Semnalele externe ale microprocesorului I8086 sunt prezentate în figura 4.6.

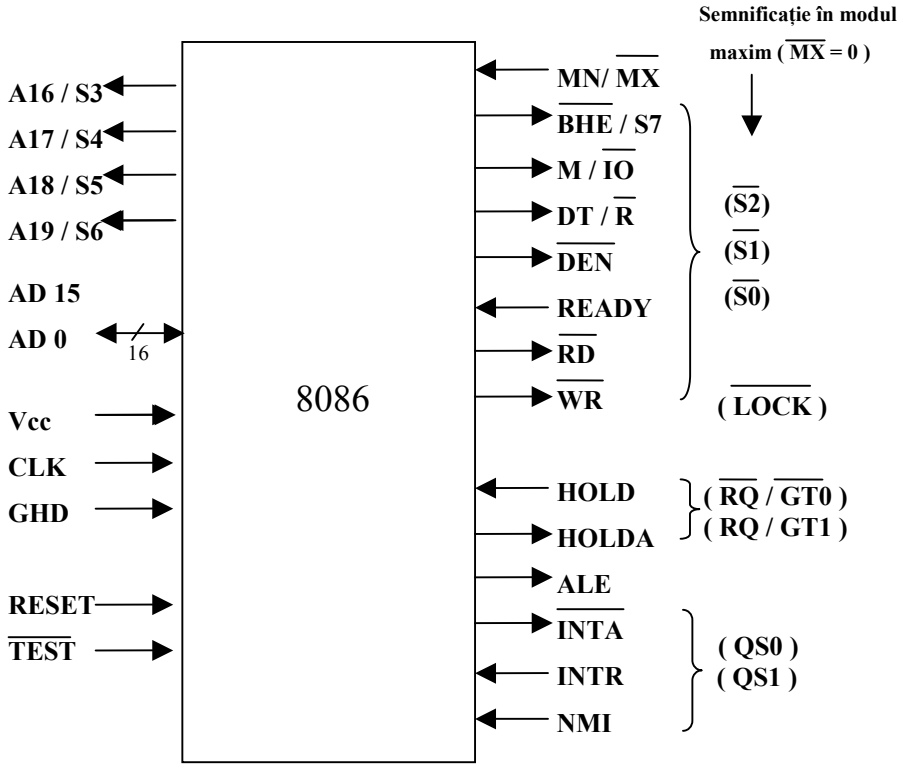


Figura 4.6. Semnalele externe ale microprocesorului I8086.

În tabelul 4.1 sunt date semnificaţiile semnalelor externe ale microprocesorului I8086 atât în modul "minim" cât şi cel "maxim". Microprocesorul I8086 este realizat în capsule de 40 de pini. Din acest motiv semnalele acestuia sunt multiplexate, în aşa fel

încât să poata fi generate toate semnalele pe aceşti pini (adrese – 20 de semnale, date – 16 semnale, comenzi şi alimentare), cele 16 linii de adresă A15 … A0 fiind multiplexate cu cele 16 linii de date D15 … D0.

TABELUL 4.1.

	Simbol	Nr. pin	Semnificație	Tip
SEMNIFICAȚII FIXE	AD15 – AD0	2÷16, 39	Adresa în T1/D este în T2, T3, Tw, T4	I/O, Z
	A19 / S6	35 ÷38	Adresa în T1 / semnele de stare în T2, T3, Tw, T4	
	A18 / S5			
	A17 / S4			
	A16 / S3			
	MN / MX	33	Comanda modului: 1 minim; 0 maxim	I
	BHE / S7	34	Validare bus (HIGH) în T1 / stare în T2 ÷ T4	**O, Z**
	RD	32	Comandă de citire pe bus-ul local	O, Z
	READY	22	Memorie / (I/E) "goto"pentru transfer	I
	RESET	21	Comanda de resetare	I
	TEST	23	Testată de instrucțiuni WAIT (Aşteaptă până ce TEST = 0	I
	INTR	18	Întreruperi mascabile externe; activă pe nivel	I
	NMI	17	Întreruperi nemascabile externe; activă pe front (+)	I
	CLK	19	Semnal de test generat de 8284	I
	VCC	40	Alimentare +5V	I
	GHN	1, 20	Masă	

SEMNIFICAȚII ÎN MOD MINIM	M / IO	28	Ieşire de stare. Selecție memorie (I / E)	O, Z
	WR	290	Comandă de scriere	O, Z
	DT / R	27	Comandă semnul transferului; 1=> transmisiune; 0 => recepție	O, Z
	DEN	26	Validare date. Activ în cicluri M, I/E, INTA	O, Z
	ALE	25	Indică prezența adresei pe AD0 ... AD15	**O**
	INTA	24	"READ"pentru cele două cicluri INTA ale lui 8086	O
	HOLD	31	Cerere de bus de la alt "master"	I
	HLDA	30	Confirmare de cedare a bus-ului	O
SEMNIFICAȚII ÎN MOD MAXIM	S2, S1, S0	26 ÷ 28	Semnale de stare pentru ciclu de bus	O
	RQ / GT1	30	Comenzi cerere/cedare pentru arbitraj de bus	I / O
	RQ / GT0	31		
	LOCK	29	Indicator că 8086 nu va ceda bus-ul altui "master"	O
	QS1 – QS0	24, 25	Starea cozii de instrucțiuni din interfața cu bus-ul a UCP	O

unde: I = intrări; O = ieşiri; Z = impedanță mare;

Observaţii:

- cu T_n se notează ciclurile unității centrale (T_1 – ciclul de extragere cod operație etc.)
- AD15 – AD0 : generează adrese în intervalul T1, devin intrări / ieşiri de date în T2 ÷ T4 şi trec în impedanță mare în timpul ciclurilor INTA, sau dacă UCP a cedat bus-ul (HDLA = 1);
- A19 / S6 – A16 / S3: generează adrese în intervalul T1 (cu valoare "0"dacă ciclul de transfer este I/E) şi semnale de stare în T2 ÷ T4.
  - S6 = 0 indică ocuparea bus-ului de către UCP;
  - S5 = IF copiează starea flag-ului de intrerupere. Astfel, starea de validare / invalidare se poate citi hardware din exteriorul UCP.

S4	S3		
0	0	"Alternate data"	Indică registrul segment curent utilizat pentru adresare.
0	1	"Stack"	
1	0	"Code or home"	
1	1	"Data"	

Aceşti pini trec în starea Z în timpul cât HDLA = 1

- BHE şi A0 determină tipul transferului pe 8 sau 16 biți, astfel:

BHE	A0	tipul transferului
0	0	16 biți
0	1	MSB (adresă impară)
1	0	LSB (adresă pară)
1	1	-

BHE = 0 atunci când un octet trebuie transferat pe cei 8 biți mai semnificativi (MSB) ai bus-ului cu 16 biți. El este activ în timpul ciclurilor "RD", "WR", "INTA" (este "0" în timpul primului ciclu "INTA").

BHE trebuie demultiplexat odată cu A0 ÷ A15 (prin memorare pe frontul negativ al semnalului I/E deoarece în T2 ÷ T4 el indică bitul de stare S7.

În modul "maxim", o parte din semnificațiile pinilor se modifică.

- S2, S1, S0 – identifică tipul de ciclu maşină în intervalele T4, T1, T2, informând controlerul de sistem I 8086 asupra tipului de transfer care urmează (ele sunt inactive în T3 şi Tw).

  S2, S1, S0
  0 – confirmarea acceptării întreruperii,
  1 – citire I/E,
  3 – HALT (oprire),
  4 – Citirea codului instrucțiunii,
  5 – Citire din memorie a unui operand,
  6 – Scriere în memorie a unui operand,

7 – Ciclu inactiv.
Pinii S2, S1, S0 trec în starea Z în intervalele în care UCP cedează bus-ul.

- Liniile bidirecționale RQ / GT, (RQ/GT0 au prioritate față de RQ/GT1) şi sunt utilizate de alte module "master"de pe bus-ul local pentru pentru a cere bus-ul de la UCP. Cererea se face prin RQ = 0 (intrare); la sfârşitul ciclului maşină curent UCP cedează bus-ul şi generează GT = 0 ("GranT"). Semnalul LOCK = 0 indică faptul că UCP nu va ceda bus-ul deoarece execută o secțiune de transfer neîntreruptibilă ce trebuie terminată.
- Semnalele QS0 şi QS1 indică starea cozii de instrucțiuni existente în unitatea de interfață de bus a UCP.

Firma Intel a creat o serie de circuite integrate necesare realizării microsistemelor cu microprocesor I8086. O parte din aceste circuite au devenit modele pentru dezvoltările ulterioare, foarte multe din circuitele moderne fiind compatibile cu aceste circuite. Principalele circuite dezvoltate de firma Intel sunt :

- 8086 (IAPX 86/10), IAPX 186, IAPX 286 – UCP cu 16 biți;
- 8087 (IAPX 86/20) – procesor aritmetic în virgulă mobilă;
- 8088 (IAPX 88/10) – UCP cu 16 biți în interior şi 8 biți în exterior;
- 8089 UCP specializat în operații I/E;
- IAPX 86/30, IAPX 88/30 – procesor specializat conținând un sistem de operare în timp real încorporat în HW;
- 8284 – generator de tact;
- 8288 – controler de sistem (generează semnalele de comandă pe bus-uri);
- 8289 – arbitru de bus (coordonează funcționarea mai multor UCP-uri pe acelaşi bus) pentru sisteme multiprocesor;
- 8259A – controler de întreruperi;
- 8237A – controler DMA;
- 8282, 8283 – circuite lactch (8biți);
- 8286, 8287 – circuite buffer (8biți).

În figura 4.7 este prezentat un microsistem realizat cu microprocesorul I8086 conectat în modul "minim".

Din această figură se observă existența celor trei magistrale : magistrala de comenzi, magistrala de adrese şi magistrala de date. Magistrala de adrese şi cea de date, datorită faptului că sunt multiplexate, necesită circuite suplimentare pentru generare : circuitul I8282 latch şi circuitul I8286 buffer.

În figura 4.8. este prezentat modul în care microprocesorul I8086 poate fi conectat în modul "maxim". Aşa cum s-a arătat, în acest mod sunt generate în mod suplimentar semnale de comandă a magistralelor cu ajutorul circuitului controler de sistem I8288.

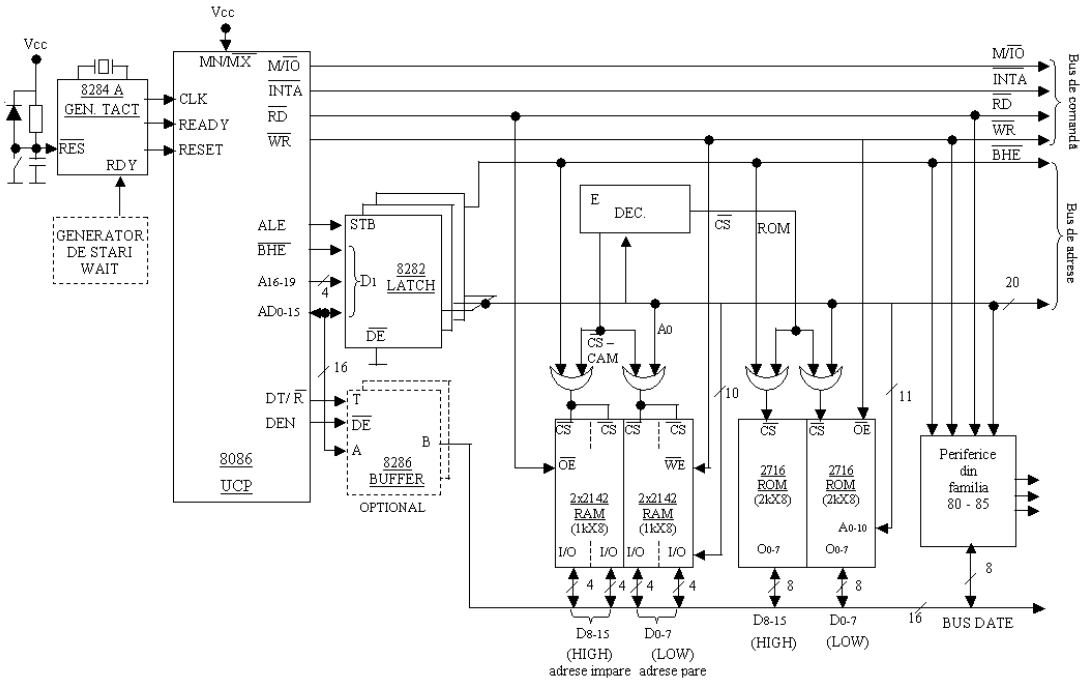


Figura 4.7. Microsistem cu microprocesor I8086 conectat în modul "minim".

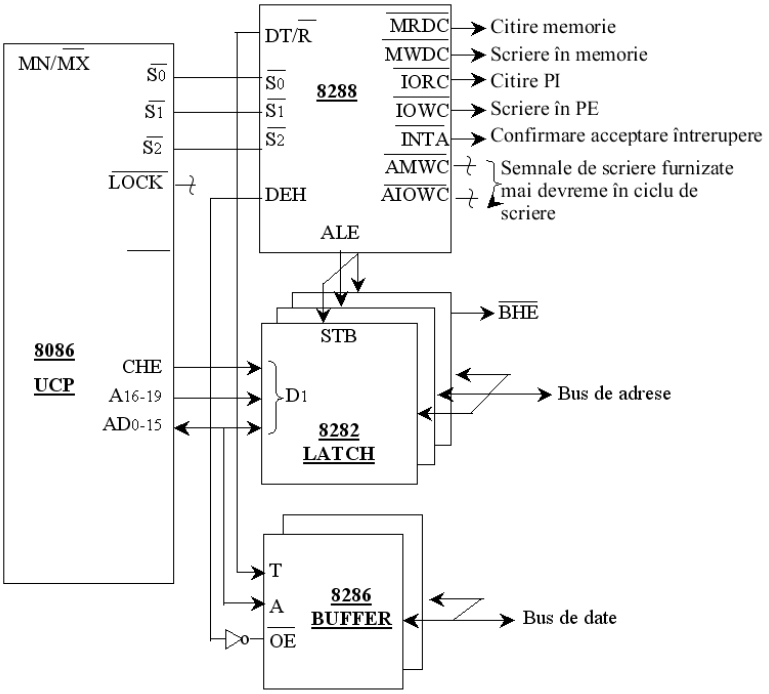


Figura 4.8. Conectarea microprocesorului I8086 în modul "maxim"

Organizarea memoriei principale

Microprocesorul poate adresa direct 1Mo de memorie la adresele 0 ÷ 0FFFFFH.

Tipuri de date memorate:

- 8 biți (octeți)
- 16 biți (cuvinte)
- 32 biți (cuvinte duble sau pointer-i)

Nu există restricții privind plasarea acestor date în memorie (ele pot "începe"la orice adresă). În funție de plasarea datelor la scheme pare sau impare UCP va executa automat numărul de cicluri necesare pentru citirea lor.

Convenția de reprezentare a datelor multi-cuvânt în memorie este cea standard la microprocesoarele INTEL: octeții mai puțin semnificativi sunt plasați la adrese mai mici (în grupul de octeți alocat cuvântului).

Memoria lui I 8086 este segmentată.

Pentru generarea adresei fizice AF, BIU execută operația dată de relația:

$$AF = S \times 2^4 + O \qquad (4.1)$$

S = conținutul registrului segment

O = adresa efectivă (offset)

Se impune deci restricția ca un segment să înceapă la o adresă absolută multiplu de 16.

Registrele segment pot fi implicite sau explicite (la dorința programatorului):

Tipul de referire la memorie	Registrul segment utilizat implicit	Alte registre utilizate	OFFSET (adresa relativă în cadrul segmentului)
"FETCH"	CS	_	IP
* Operații cu stiva	SS	_	SP
* Date variabile (cu excepțiile de mai jos)	DS	CS, SS, ES	Adresa efectivă
* Sursa la operațiile cu şiruri	DS	CS, SS, ES	SI
* Destinația la operațiile cu şiruri	ES	_	DI
* BP folosit ca registru de bază	SS	CS, DS, ES	Adresa efectivă

Utilizarea altor registre segment față de cele utilizate implicit trebuie indicată de programator în instrucțiunea respectivă cu ajutorul unor prefixe speciale (1 octet).

Memoria stivă

Stiva sistemului se organizează în memoria principală. Într-o aplicație pot exista mai multe stive (fiecare cu dimensiunea maximă de 64Ko). Adresa de bază a stivei (valoarea inițială a registrului SP într-un program) este diferită de adresa de bază a segmentului de memorie alocat stivei.

Modul de organizare al stivei microprocesorului I8086 este arătat în figura 4.9.

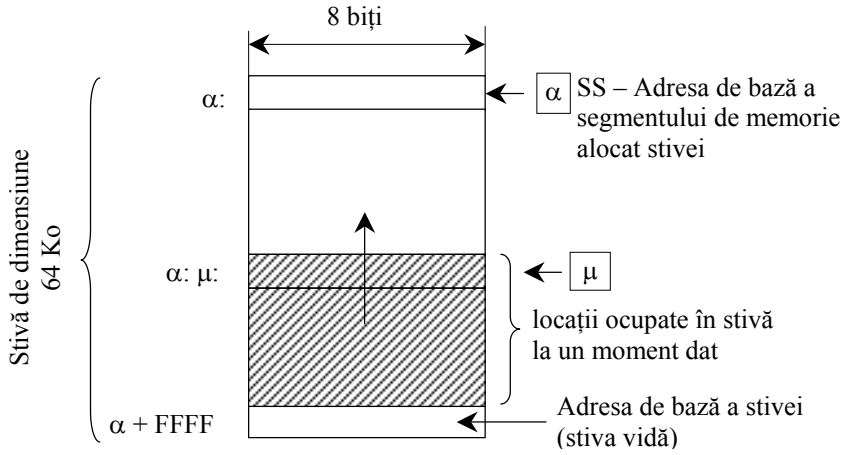

8 biți
α:
α
SS – Adresa de bază a segmentului de memorie alocat stivei
Stivă de dimensiune 64 Ko
α: μ:
μ
locații ocupate în stivă la un moment dat
α + FFFF
Adresa de bază a stivei (stiva vidă)

Figura 4.9. Organizarea stivei microprocesorului 8086.

Anumite zone din memoria principală a sistemului sunt rezervate pentru diferite utilizări de către unitatea centrală.

Adrese rezervate:

- 0 ÷ 13H = dedicate pentru întreruperi interne;
- 14H ÷ 7FH = rezervate pentru dezvoltări ale familiei de componente (rezervate de firma INTEL);
- 0FFFF0H ÷ 0FFFFBH = dedicate pentru instrucțiunile executate resetarea procesului;
- 0FFFFC ÷ 0FFFFFH = rezervate de firmă

Porturile de intrare / ieşire (I / E)

Spațiul de adrese de intrare / ieşire este separat de spațiul adreselor memoriei. Spațiul este nesegmentat şi cu dimensiunea de 64 Ko ceea ce asigură posibilitatea de adresare directă a unui mare număr de porturi I/E cu 8 sau 16 biți.

Într-un transfer I/E, UCP citeşte/scrie 16 biți/ciclu de bus dacă portul I/E este localizat la adresa pară şi 8 biți/ciclu dacă portul I/E este localizat la adresă impară.

Facilități pentru lucru în sisteme multi-master

Facilitățile pentru lucrul multi-master sunt prezentate în continuare.

- Interzicerea accesului la bus a altor module master cu ajutorul semnalului LOCK. Bus-ul se utilizează prin diviziune în timp,

întreţeserea ciclurilor de acces la bus a diverselor module "master" se face la nivel de ciclu de bus (şi nu la nivel de ciclu de instrucţiune).
LOCK = 0 indică faptul că 8086 execută o instrucţiune ce nu poate fi întreruptă (instrucţiunea e prefixată "LOCK" – ex. actualizarea unui pointer de 4 octeţi).

- Utilizarea liniilor RQ/GT0 şi RQ/GT1
  Liniile RQ/GT0 şi RQ/GT1 bidirecţionale permit controlul accesului la un bus local comun mai multor procesoare. Protocolul de cerere/cedare a bus-ului decurge astfel:
  - procesorul solicitator cere acces prin generarea unui impuls (RQ=0) către UCP;
  - UCP (la sfârşitul ciclului maşină curent) cedează bus-ul şi răspunde prin impulsul (GT=0);
  - după terminarea accesului, celălalt procesor eliberează bus-ul informând asupra acestui fapt printr-un nou impuls (GT=0).

Sistemul de întreruperi

- Clasificarea întreruperilor
  Întreruperile interne (software)

  - INT n, unde tipul este TIP=n, n ∈ {0, ..., 255}
  - INTO (INTerrupt ou Overflow), provoacă o întrerupere TIP=4 dacă flag-ul overflow a fost setat în urma apariţiei unor depăşiri la efectuarea operaţiilor aritmetice;
  - IDIV, cu TIP=0, generată automat dacă apare o eroare de depăşire la operaţia de împărţirre;
  - SINGLE STEP, cu TIP=1, se generează automat după execuţia fiecărei instrucţiuni, dacă flag-ul TF ("test flag") a fost setat în "1". Acest tip de întrerupere practic UCP în modul de lucru "pas cu pas", util pentru depanarea programelor.

Întreruperile externe (hardware) se generează prin aplicarea unor semnale pe intrările de întrerupere.

  - NMI – pentru întreruperile nemascabile, cu TIP=2; semnalul este activ pe frontul pozitiv.
  - INTR – pentru întreruperi mascabile (prin flag-ul IF din registrul de flag-uri). Semnal activ prin nivel logic 1; acesta trebuie menţinut activ până la recunoaşterea întreruperii de către UCP. Tipul întreruperii externe mascabile se generează de către dispozitivul întrerupător (de exemplu controlerul de întreruperi I8259A).

- Servirea întreruperilor

Tipul unei întreruperi este utilizat de către UCP ca adresă relativă într-un tablou de "pointer"-i, amplasat în memoria principală între adresele 0÷3FFH (256x4 octeţi).

Un pointer conține adresa logică pentru servirea întreruperii. Pentru a efectua saltul la această adresă se efectuează următoarele operații:

IP ← TP(T+1,T); CS←TP(T+3, T+2)
T=4xTIP

TP(α+1, α) reprezintă cuvântul cu 16 biți format din octeții de adresă α-1 şi α din tabloul de "pointer"-i. Efectul este asemănător cu cel al instrucțiunii CALL intersegment

Prioritățile de servire pentru diversele tipuri de întreruperi (care pot apare eventual simultan) rezultă din organigrama alăturată.

Ciclurile "INTA" pentru întreruperile externe mascabile (INTR) – pe durata celor două cicluri INTA, semnalul LOCK este activ. Vectorul de întrerupere este în acest caz numărul indicator al tipului şi este citit de către UCP în al doilea ciclu "INTA".

În figura 4.10 se prezintă modul de alocare în memorie a vectorilor de întrerupere.

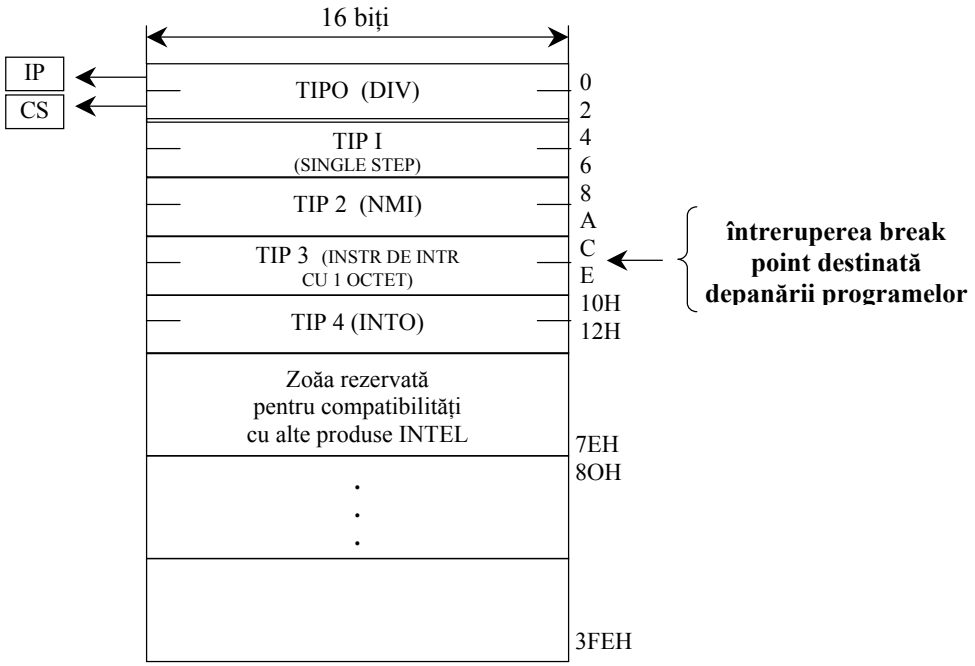


Figura 4.10. Tabel de "pointer"-i pentru întreruperi

În figura 4.11 se prezintă organigrama corespunzătoare servirii unei întreruperi de către unitatea centrală I8086.

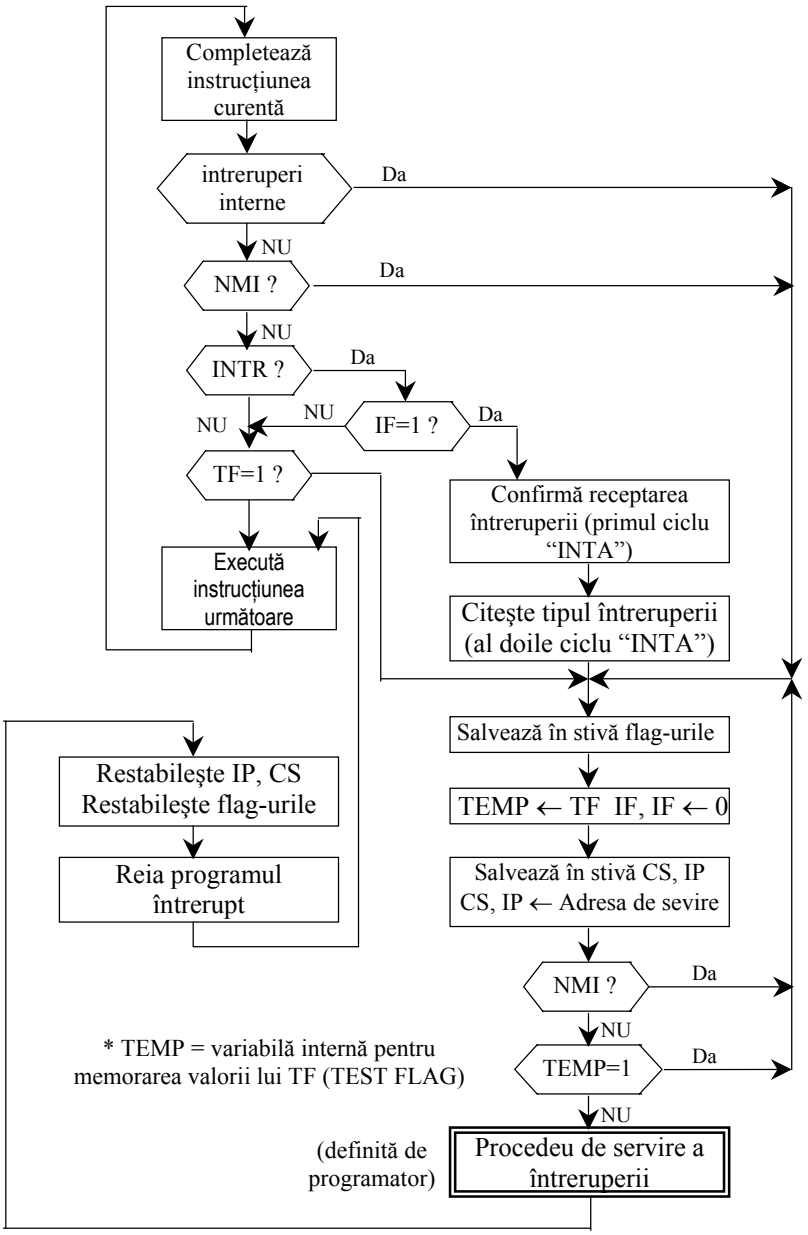


Figura 4.11. Organigrama servirii unei întreruperi.

Inițializarea unității centrale. Starea HALT. Utilizarea intrării TEST

- 1 pe RESET determină:

CS ← FFFFH, IP ← 0 ; ceea ce determină generarea adresei fizice 0FFFF0H cu adresă absolută a primei instrucțiuni executată de I8086.

Intreruperile mascabile se invalidează IF ← 0.

- HALT (la apariția instrucțiunii) stopează toate activitățile UCP până ce apare o întrerupere sau semnal RESET. În această sare o cerere de bus este (HOLD=1) este recunoscută şi şi acceptată de către UCP.
- Intrarea TEST este testată de instrucțiunea WAIT şi dacă TEST ≠ 0, atunci procesorulaşteaptă (într-o stare inactivă) până ce TEST ← 0. În acest timp, se pot servi întreruperile apărute la intrări, după care se reia aşteptarea. Intrarea TEST poate servi pentru sincronizarea UCP cu evenimente externe.

4.4. Procesoare de semnal digitale

O dată cu dezvoltarea tehnicii digitale şi în special cu creşterea performamțelor microprocesoarelor s-a dezvoltat un segment aparte de componente numerice specializate în prelucrarea numerică a semnalelor. Asfel de componente au căpătat numele de procesoare de semnal digitale, prescurtat DSP (Digital Signal Processor). Astfel prescurtarea DSP capătă d dublă semnificație; una din semnificații se referă la tehnica prelucrării semnalelor în format numeric iar cea de-a doua semnificație se referă la dispozitivele (procesoarele) specializate, destinate implementării tehnicilor de prelucrare a semnalelor în format numeric.

Un procesor destinat prelucrării numerice a semnalelor reprezintă o unitate centrală specializată care este capabilă să execute cu viteză ridicată secvențe de instrucțiuni cum sunt cele de deplasare a conținutului unui registru şi adunarea conținutului acestuia sau cele de înmulțire şi adunare care sunt operații uzuale în algoritmii de prelucrare a semnalelor.

Spre deosebire de un microprocesor care este o unitate centrală de uz general, un dispozitiv DSP este destinat anumitor domenii de utilizare definite de tehnica de prelucrare numerică a semnalelor având instucțiuni speciale, adaptate scopului propus iar aplicatiile sunt executate în timp real ceea ce presupune untimp scurt de execuție a instrucțiunilor şi o structură specială a sistemului de întreruperi. Pe de altă parte, un dispozitiv DSP lucrează de obicei într-o structură ierarhizată de calcul în care dispozitivul DSP asistă un microprocesor de uz general.

Deşi există o mare varietate de dispozitive DSP acestea sunt proiectate cel mai adesea să îndeplinească aceleaşi funcții de bază. Rezultă că există un set de caracteristici de bază pentru toate procesoarele DSP, caracteristici de bază care pot fi împărțite în trei categorii:

- dispozitivele sunt specializate în efectuarea cu viteză ridicată a operațiilor aritmetice;
- sunt prevăzute cu mecanisme diversificate de transfer a datelor din şi către lumea reală;
- sunt utilizate arhitecturi cu acces multiplu la memorie.

Activitățile unui dispozitiv DSP impun efectuarea unor operații specifice cum sunt: (figura 4.12):

- adunări şi înmulțiri;

- întârziri;
- manipulari de matrici.

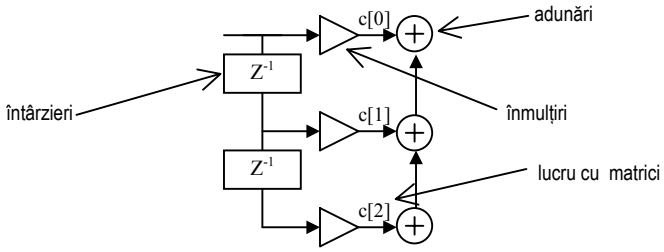


Fig. 4.12. Operații specifice prelucrării digitale a semnalelor

Operațiile de adunare şi de înmulțire sunt folosite pentru:

- extragerea simultană a doi operanzi;
- realizarea adunării şi înmulțirii (de obicei simultan);
- stocarea rezultatului sau reținerea acestuia în vederea repetării operației.

Întârzierile sunt folosite pentru reținerea unei valori pentru utilizare ulterioară.
Manipularea matricilor este necesară pentru:

- extragerea operanzilor aflați în locații sucesive de memorie;
- copierea datelor de la memorie la memorie.

Această suită de operații sunt de obicei executate de dispozitivul DSP astfel:

- executarea în paralel a operațiilor de înmulțire şi adunare;
- acces multiplu la memorie în scopul extragerii simultane a doi operanzi şi memorarea rezultatului;
- folosirea mai multor regiştrii pentru reținerea temporară a datelor;
- generarea eficientă a adreselor pentru utilizarea matricilor;
- facilități speciale cum sunt întârzierile sau adresarea circulară.

Pentru realizarea operațiilor aritmetice dispozitivele DSP au o structură specială, de mare viteză, a unității logice şi aritmetice deoarece ele trebuie să execute simultan adunări şi înmulțiri. Din acest motiv dispozitivele DSP au de regulă o structură specială a circuitelor de adunare şi înmulțire ce permite acestor operații în paralel, de către o singură instrucțiune, aşa cum este arătat în figura 4.13.

Întârzierea permite ca valoarea intermediară rezultată în urma calculelor să fie memorată pentru o utilizare ulterioară. Acest lucru poate fi necesar, de exemplu, când trebuie calculat un total; totalul poate fi păstrat în procesor pentru a evita scrierile şi citirile repetate la memorie. Din acest motiv procesoarele DSP au un număr relativ mare de regiştrii, în virgulă fixă sau virgulă mobilă, care pot fi folosiți pentru stocarea valorilor intermediare.

Utilizarea matricilor permite ca datele să poată fi manevrate eficient în locații succesive de memorie. Acest lucru necesită generarea adresei de memorie unde se găseşte valoarea următoare şi pentr aceasta dispozitivele DSP au regiştrii de adrese ce sunt folosiți pentru păstrarea adresei şi care permit generarea adresei următoare în mod eficient.

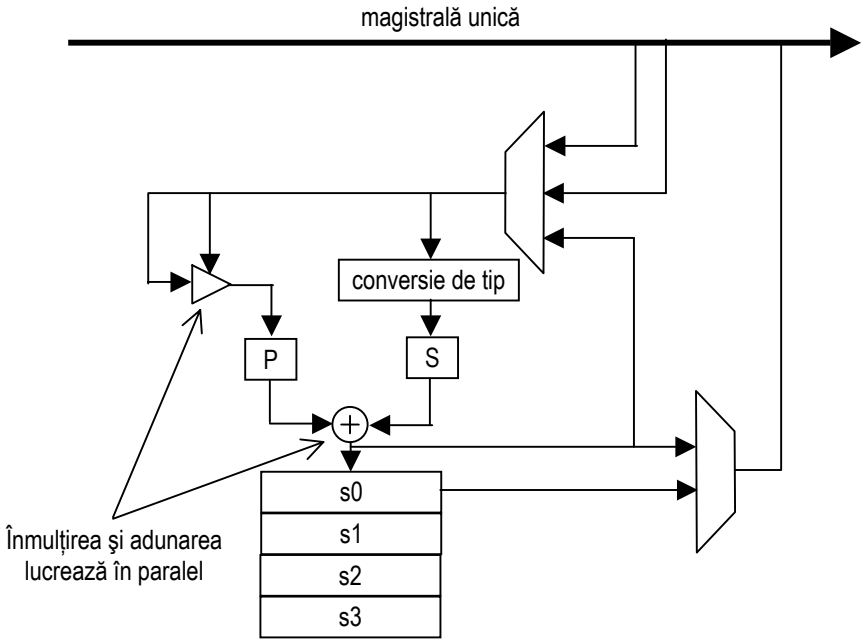


Fig. 4.13. Fluxul de date la realizarea inmulțirii şi adunării

TABELUL 4.2.

Simbol	**Adresare**	**Observații**
*rP	adresare indirectă prin registru	este citită data din memorie indicată de adresa conținută în registrul rP
*rP++	postincrementare	după citirea datei din memorie, adresa din registrul rP este incrementată, în aşa fel încât să indice adresa următoarei date din matrice
*rP--	postdecrementare	după citirea datei din memorie, adresa din registrul rP este decrementată, în aşa fel încât să indice adresa următoarei date din matrice
*rP++rI	postincrementare în funcție de registru	după citirea datei din memorie, adresa conținută în registrul rP este incrementată cu valoarea conținută de registrul rl, adresa obținută indicând noua valoare ce urmează a fi citită din memorie further down the array
*rP++rIr	cu biți inversați	după citirea datei din memorie, valoarea ce indică adresa este incrementată pentru a indica valoarea următoare din matrice, biții adresei fiind aşezați în ordine inversă

Posibilitatea generării eficiente a unei adrese noi este o facilitate caracteristică a procesoarelor DSP. În mod obişnuit adresa următoare poate fi generată pe durata

extragerii sau stocării datelor. Procesoarele DSP au un set bogat de instrucțiuni pentru generarea adreselor. Un exemplu de astfel de instrucțiuni este dat în tabelul 4.2.

Sintaxa în limbaj de asamblare pentru instrucțiunile prezentate în tabelul 4.2 este foarte asemănătoare cu limbajul C. De câte ori un operand este extras din memorie prin utilizarea modului de adresare indirect prin regiştrii, registrul de adresă poate fi incrementat pentru a indica următoarea valoare necesară din matrice. Incrementarea adresei este liberă – nu sunt implicate resurse pentru calculul adresei – şi mai multe astfel de adrese pot fi generate într-o singură instrucțiune. Generarea adreselor este un factor important în creşterea vitezei procesoarelor DSP şi a operațiilor specializate ale acestora.

Ultimul mod de adresare – cu biți inversați – arată cât de specializate pot fi procesoarele DSP. Adresarea cu inversare de biți este necesară atunci când un tabel de valori este reordonat prin inversarea ordinii biților de adresă astfel:

- inversarea ordinii biților în fiecare adresă;
- amestecarea datelor prin inversarea biților, adresarea făcându-se în ordine crescătoare.

O astfel de operație este utilizată exclusiv la calculul tansformatei Fourier rapide. Se poate spune deci, că dispozitivele DSP sunt proiectate special pentru a calcula în mod eficient transformata Fourier rapidă.

Din punct de vedere al legăturii unității centrale cu mediul extern, un dispozitiv DSP este prevăzut cu mai multe interfețe interne (on chip) care lucrează într-un sistem specializat de întreruperi. Cele mai frecvente interfețe ale unui dispozitiv DSP sunt:

- intrări/ieşiri numerice;
- convertoare analog-numerice (CAN);
- convertoare numeric-analogice (CNA);
- comparatoare;
- numărătoare programabile;
- interfațe seriale sincrone de mare viteză pentru conectarea perifericelor externe sau a dispozitivelor DSP;
- interfețe seriale standard de tip RS-232 pentru lucrul pe port serial asincron.

4.4.1. Procesorul de semnal digital, TMS320F240

Circuitul DSP de tip TMS320F240 este un circuit integrat pe scară foarte largă în tehnologie CMOS şi se compune din trei unități funcționale: unitatea centrală de tip C2xx DSP, memoria internă şi unitatea circuitelor periferice. De asemenea, în afară de aceste unități funcționale sunt prevăzute o serie de facilități sistem care se referă la gestionarea memoriei, initializarea blocurilor, intreruperi, controlul intrărilor/ieşirilor, generearea ceasului intern şi comutarea în regim de consum redus. Circuitul TMS320F240 utilizează o arhitectură Harvard avansată pe folosirea a două magistrale separate: de date şi de program.

În continuare sunt prezentate foarte sumar principalele facilități oferite de circuitul DSP în scopul formării unei imagini asupra structurii complexe a acestuia:

- unitate centrală de tip C2xx cu arhitectură paralelă ce oferă posibilitatea execuției în paralel a instrucțiunilor, posibilitatea de prelucrare a mai multor fluxuri de informații simultan, execuția instrucțiunilor într-un singur ciclu maşină (inclusiv a inmulțirii);
- memorie de program inclusă în circuitul integrat de 16KB (cuvinte de 16 biți) de tip flash EEPROM şi memorie RAM cu acces dublu pentru program/date de 544 cuvinte de 16 biți;
- oscilator extern de 10MHz, bucla PLL internă a DSP lucrând la 20MHz;
- 28 de intrări/ieşiri numerice;
- două convertoare analog-numerice pe 10 biți ce pot lucra simultan, numărul intrărilor analogice fiind extins prin multiplexare la 16;
- trei numărătoare programabile independente pe 16 biți;
- interfață serială sincronă de mare viteză pentru conectarea perifericelor externe;
- interfață serială standard de tip RS-232 pentru lucrul pe port serial asincron;

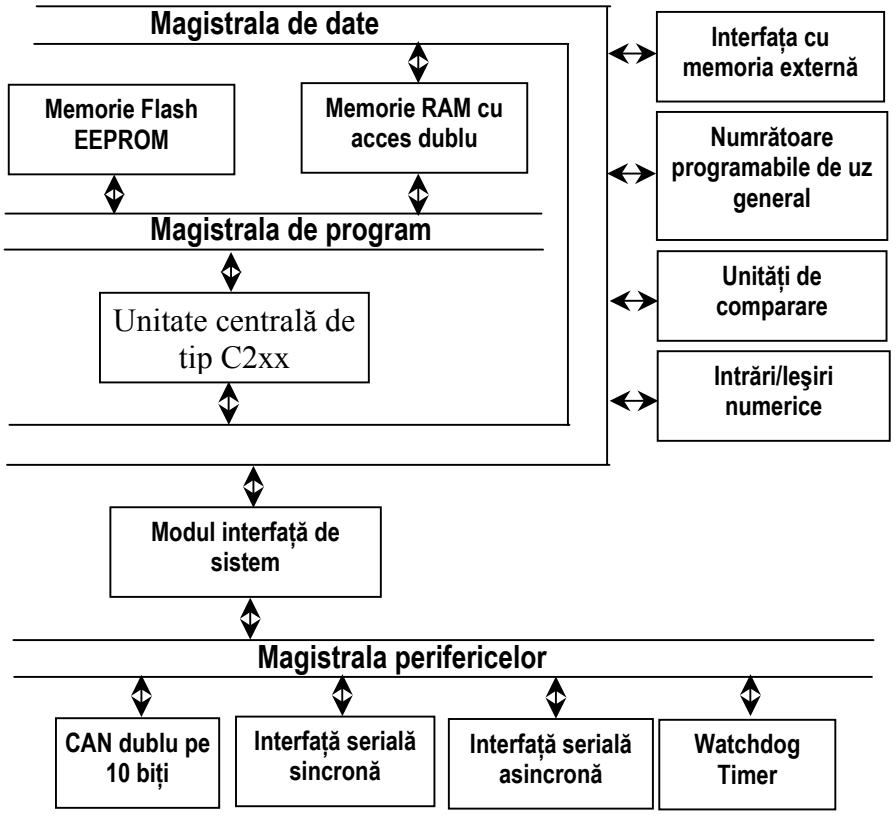


Fig. 4.14. TMS320F240 schema bloc

Schema bloc simplificată a circuitului DSP de tip TMS320F240 este prezentată în figura 4.14.

Prezența celor două magistale, magistrala de date şi magistrala de program permite prelucrarea simultană a instrucțiunilor de program şi a datelor.

Organizarea memoriei procesorului este prezentată în tabelele 4.3, 4.4 şi 4.5.

TABEL 4.3.

Memorie Program

Hex	
0000 003F	Intreruperi (Memorie externă)
0040 FDFF	Memorie externă
FE00 FEFF	On-Chip DARAM B0 (CNF=1) sau **Memorie externă** (CNF=0)
FF00 FFFF	Rezervat

MP/$\overline{MC}$ =1

Mod microprocesor

TABEL 4.4.

Memorie Program

Hex	
0000 003F	Intreruperi On-Chip Flash
0040 3FFF	Memorie Flash On-Chip
4000 FDFF	Memorie externă
FE00 FEFF	On-ChipDARAM B0 (CNF=1) sau **Memorie externă** (CNF=0)
FF00 FFFF	Rezervat

MP/$\overline{MC}$ =0

Mod Microcomputer

Dispozitivul DSP este prevăzut cu un pin MP/$\overline{MC}$ (microprocessor mode / microcomputer mode) cu ajutorul căruia se pot obține două configurații ale memoriei de program:

- MP/$\overline{MC}$ = 1 (microprocessor mode) modul în care memoria on chip flash şi ROM sunt dezactivate (Tabelul 4.3);
- MP/$\overline{MC}$ = 0 (microcomputer mode) modul în care memoria on chip flash şi ROM sunt activate (Tabelul 4.4).

Bitul CNF care se găseşte în registrul de stare ST1 permite activarea memoriei RAM cu acces dublu on chip, blocul 0 (CNF = 1) sau a memoriei externe (CNF = 0).

Organizarea memoriei de date este arătată în tabelul 4.5. Se observă că în memoria de date vom găsi corespondentul regiştrilor unității centrale şi a perifericelor ceea ce permite lucrul cu aceştia identic cu lucrul cu orice altă locație de memorie. Acest fapt duce la creşterea vitezei de execuție şi simplitatea programării.

Împărțirea memoriei în două tipuri: memorie de program şi memorie de date impune ca orice program să fie alcătut din două segmente corespunzătoare acestor tipuri, amestecarea instrucțiunilor cu datele nefiind permisă.

TABEL 4.5.
Memorie Date

Hex	
0000 005F	Regiştrii şi zonă rezervată
0060 007F	Memorie On-Chip DARAM B2
0080 01FF	Rezervat
0200 02FF	Memorie On-Chip DARAM B0 (CNF=0) sau zonă rezervată (CNF=1)
0300 03FF	Memorie On-Chip DARAM B1
0400 07FF	Rezervat
0800 6FFF	Ilegal
7000 73FF	Regiştrii perifericelor (System, WD, ADC, SPI, SCI, Intreruperi, I/O)
7400 743F	Regiştrii perifericelor (Manager-ul de evenimente)
7440 77FF	Rezervat
7800 7FFF	Ilegal
8000 FFFF	Memorie externă

Organizarea întreruperilor unității centrale

Sistemul de întreruperi externe ale unității centrale (miezul procesorului C2xx - CPU) este format din şase întreruperi mascabile (INT1-INT6) şi una nemascabilă (NMI). Prioritatea maximă o are întreruperea NMI iar prioritățile întreruperilor mascabile scad de la INT6 la INT1, INT1 având prioritatea minimă.

Fiecărei întreruperi îi corespunde un bit în registrul fanioanelor de întrerupere a unității centrale (IFR) şi fiecărei întreruperi mascabile îi corespunde un bit în registrul măştilor unității centrale (IMR). O întrerupere mascabilă este mascată (nu va genera o întrerupere către miez) când bitul corespunzător în IMR este 0. De asemenea miezul procesorului mai are un bit pentru mascarea generală a întreruperilor (INTM) în registrul de stare ST0. Când INTM este setat la 1 toate întreruperile mascabile sunt dezactivate.

Răspunsurl la întrerupere a miezului C2xx

Când apare o tranziție de la unu la zero pe o intrare de întrerupere a miezului, bitul corespunzător al fanionului din IFR este setat în 1. O întrerupere este generată către miez dacă aceasta nu este mascată - întreruperile globale sunt permise (INTM=0) - şi nu există o altă întrerupere nemascată de prioritate mai mare în aşteptare (acest lucru însemnând că nu există nici un fanion setat al unei întreruperi nemascate de prioritate mai mare). Fanionul este şters de către hardware o dată ce cererea de întrerupere este trimisă către miez. Un fanion de întrerupere poate fi de asemenea şters de către programul utilizator scriind un 1 în bitul corespunzător.

4.5. Microcalculatoare integrate, microcontrolere

4.5.1. Prezentare generală

Electronica digitală se bazează astăzi în bună măsură pe utilizarea circuitelor integrate VLSI (Very Large Scale Integration – circuite integrate pe scară foarte largă) de tipul microprocesor, microcalculator sau microcontroler.

Dintre acestea se detaşează net microcalculatoarele integrate şi microcontrolerele care înglobează în structura hard pe lîngă o unitate centrală puternică cu facilităţi speciale pentru lucrul în timp real şi periferice de tipul convertoarelor A/D, interfeţe paralele, interfeţe seriale, sisteme de generare a impulsurilor modulate, timere, circuite de supraveghere a duratei de execuţie a programelor, executiv de timp real integrat etc. ceea ce uşureaza mult munca proiectantului şi duce la creşterea substanţială a performanţelor sistemului.

Nu exista o diferenţă netă între microcalculatoare integrate şi microcontrolere. Microcontrolerele prin setul de instrucţiuni mai redus, sunt mai intim legate de aplicaţiile de control, urmărire şi automatizare industrială.

Între microcalculatoarele integrate mai cunoscute se pot enumera: INTEL 8048/8035, MOTOROLA 6801 şi 6805 şi familia ZILOG Z8.

INTEL 8048 a fost primul microcalculator integrat; acest dispozitiv conţine într-o capsulă de 40 de pini urmatoarele:

- o unitate centrală de 8 biţi;
- o memorie RAM de 64 octeţi;
- o memorie ROM de 1 koctet;
- un număr de 27 de linii de I/E (Intrare/iEşire);
- un oscilator pilot;
- un circuit de ceas de 8 biţi.

Setul de instrucţiuni este tributar într-o oarecare masură primului microprocesor de uz general I8080, cum de altfel este şi concepţia hardware-ului, cuprinde: operaţii cu acumulatorul, transferuri de date, salturi, chemări şi reveniri din subrutine, operaţii de intrare/ieşire şi operaţii de control cu registre şi indicatori.

MC 6801 este un microcalculator integrat de 8 biţi care s-a dezvoltat din familia microprocesorului de uz general MC6800 (MOTOROLA).

Instrucţiunile microcalculatorului integrat MC6801 sunt compatibile cu instrucţiunile microprocesorului mamă. Timpul de execuţie a fost redus şi câteva noi instrucţiuni au fost adaugate; între acestea şi instrucţiunea de înmulţire fără semn.

MC6801 poate funcţiona fie ca microcalculator de sine stătător fie ca microcalculator de uz general ce poate utiliza o memorie externă de cel mult 64ko. MC6841 include pe lângă microprocesorul propriu-zis:

- o memorie fixă (ROM) de 2 ko;
- o memorie citeşte/scrie (RAM) de 128 de octeţi;
- un număr de 29 linii de I/E;

- trei circuite de ceas programabile, de 16 biți fiecare;
- un oscilator pilot.

Familia ZILOG Z8 cuprinde o variantă standard Z8611 şi mai multe versiuni de dezvoltare. Principalele caracteristici ale acestei versiuni sunt:

- 43 de tipuri de instrucțiuni;
- 124 de registre de uz general, 4 registre de I/E, 16 registre de control şi configurare;
- 32 de linii de intrare/ieşire;
- un circuit de intrare/ieşire serie;
- 2 circuite de ceas (fiecare ceas având un divizor programabil de 8 biți şi un predivizor de 6 biți);
- 6 întreruperi vectorizate, mascabile şi aranjabile în şir de priorități;
- timp de execuție mediu a instrucțiunilor de 2,2 μs.

Dintre microcontrolerele reprezentative se pot aminti:

- microcontrolerul AVR AT90S2313 al firmei ATMEL
- microcontrolerul PIC 16F87X al firmei MICROCHIP TECHNOLOGY ;

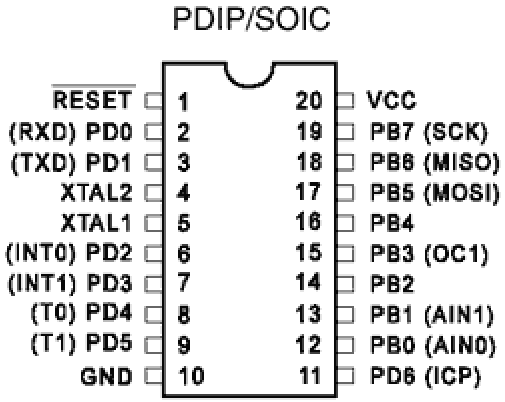
4.5.2. Microcontrolerul AT90S2313

Microcontrolerul AT90S2313 are o memorie FLASH programabila în sistem de 2K octeti.
Caracteristici principale:

- Bazat pe arhitectura AVR @ RISC cu performanțe ridicate şi consum redus:
  - 118 instrucțiuni puternice - majoritatea se execută într-un ciclu maşină;
  - 32 x 8 registre generale de lucru;
  - viteza pană la 10 MIPS @ 10 MHz.
- Memorie nevolatilă de date:
  - Mărimea memoriei Flash: 2K octeti programabilă In sistem;
    - suportă 1 000 cicli de stergere/inscriere;
  - 128 octeți de Memorie SRAM.
  - 128 octeti de memorie EEPROM programabilă In sistem;
    - suportă 1 000 000 cicli de ştergere/înscriere;.
  - Protecție programabilă pe 8 biți a memoriei de date Flash si EEPROM.
- Periferice:
  - un temporizator/numărător cu prescaler separat pe 8 biți;
  - un temporizator/numărător cu prescaler separat pe 16 biți;
  - comparator, moduri de captură şi PWM pe 8, 9 şi 10 biți;
  - comparator analogic on-chip;

  - temporizator Watchdog programabil cu oscilator încorporat in chip;
  - interfață serială SPI pentru programarea In-system;
  - port serial UART full duplex.
- Funcții speciale ale microcontrolerului:
  - consum redus în caz de inactivitate precum şi power-down de economisire a energiei;
  - surse de înrerupere externă şi internă.
- Specificatii:
  - tehnologie CMOS - consum redus, viteză mare;
  - mod de operare complet static.
- Consum de energie la 4 MHz, 3V, 25°C:
  - activ: 2.8 mA;
  - inactiv: 0.8 mA;
  - power-down: <1 μA.
- Intrări/Ieşiri şi Capsula:
  - 15 linii de intrare/iesire programabile;
  - capsula PDIP si SOIC de 20 pini.
- Tensiuni de operare:
  - 2.7 - 6.0V (AT90S2313-4);
  - 4.0 - 6.0V (AT90S2313-10).
- Viteza de operare:
  - 0 - 4 MHz (AT90S2313-4);
  - 0 - 10 MHz (AT90S2313-10).

Configurația pinilor:



Descrierea pinilor microcontrolerului:

- VCC Pin de alimentare (tensiune pozitiva).
- GND Pin de masă.
- Portul B (PB7..PB0). Portul B este un port I/O bidirectional pe 8 biți. La ieşirile terminalelor se pot configura rezistoare pull-up (legare software la plusul sursei de alimentare a rezistoarelor interne – selectabil pentru fiecare pin in parte). PB0 şi PB1 de asemenea

serversc ca intrare pozitivă (AIN0) şi intrare negativă (AIN1), pentru comparatorul analogic in chip. Portul B furnizează la ieşire un curent de 20 mA asftel încât poate fi legat direct la un afişor LED. Atunci când pinii PB0 la PB7 sunt utilizaţi ca intrări şi din exterior sunt comandaţi cu semnal digital LOW, ei vor consuma energie daca rezitenţele interne de pull-up sunt activate. Dupa un Reset pinii Portului B vor fi în starea de inaltă impedanţă chiar dacă semnalul de ceas nu este activ.

- Portul D (PD6..PD0). Portul D posedă şapte pini bidirecţionali de I/O (curent debitat de 20 mA) cu rezistenţe pull-up. Pinii portulul D intră în starea de inaltă impedanţă în cazul acţionării Resetului.

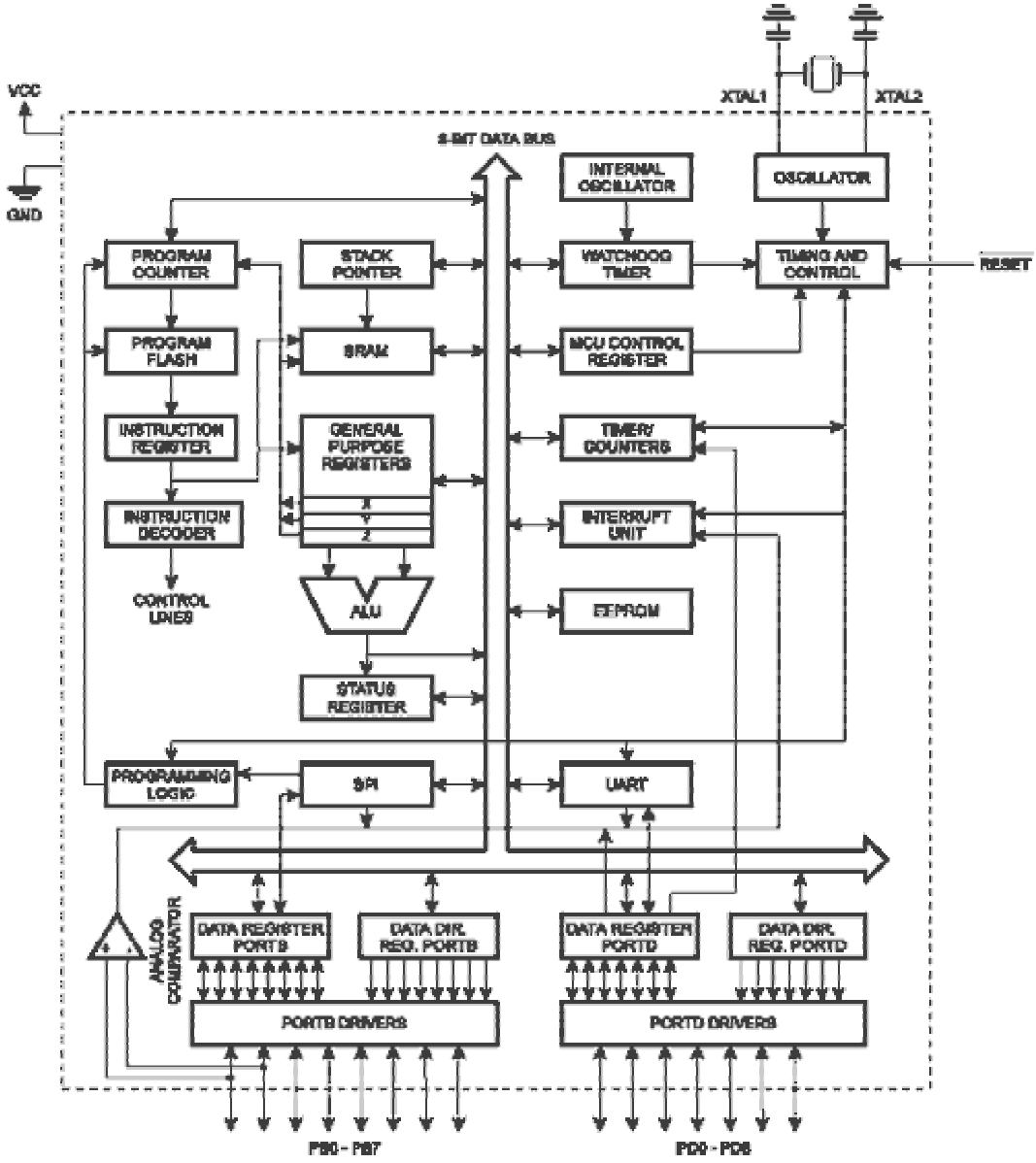


Figura 4.15. Schema bloc a microcontrolerului AT90S2313

- RESET. Pin de intrare. Un semnal LOW pe acest pin pe o durată de 50 ns va genera un Reset, chiar daca semnalul de clock nu este activ. Un semnal mai scurt nu garanteaza conditia de Reset.
- XTAL1. Intrare inversată pe un oscilator amplificat şi intrare în circuitul de semnal de ceas.
- XTAL2. Ieşire inversată de la un amplificator oscilator.

AT90S2313 este un microcontroler CMOS pe 8 biți, cu consum redus bazat pe arhitectura AVR RISC. Executând instrucțiuni puternice într-un ciclu maşină, AT90S2313 execută aproape 1 million de instrucțiuni pe secundă.

Miezul AVR combină setul bogat de instrucțiuni cu 32 de registre de lucru generale. Toate cele 32 de registre sunt conectate direct la Unitatea Aritmetică şi Logică (ALU), astfel este permisă accesarea a două registre independente într-un singur ciclu maşină. Arhitectura rezultată este mult mai eficientă din punct de vedere al codului astfel încât se obține o putere de calcul până la zece ori mai mare decât cel al arhitecturii microcontrolerelor CISC.

În modul "inactiv" procesorul se opreşte în timp ce memoria SRAM, temporizatorii/numararorii, portul SPI şi întreruperile de sistem continuă să funcționeze.

În modul "power-down" microcontrolerul salvează conținutul registrelor şi opreşte oscilatorul, se opresc toate funcțiile din chip pană la următorea întererupere externă sau un Reset hardware.

Microcontrolerul este fabricat în tehnologia memorie nevolatilă de densitate înalta. Memoria Flash internă, programabilă In sistem permite reprogramarea In sistem utilizând interfața serială SPI sau un programator de memorii nevolatile convențional.

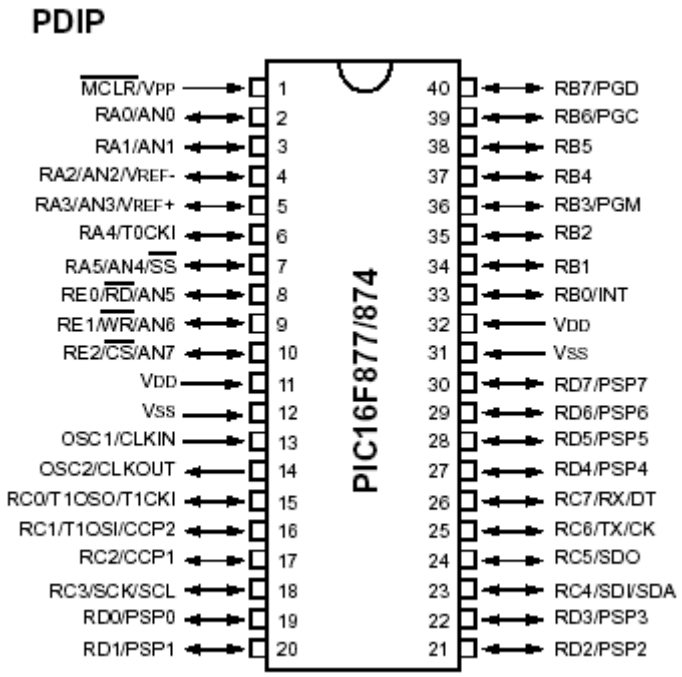
4.5.3. Microcontrolerul PIC 16F877

Microcontrolere de 28/40-Pini cu 8-Bit CMOS FLASH.
Specificațiile unității centrale:

- unitate centrala (CPU) de tip RISC;
- 35 de instrucțiuni;
- toate instrucțiunile se execută într-un ciclu al unității centrale, cu excepția instrucțiunlor de ramificare a programului care se execută în două cicluri CPU;
- frecvența de ceas: 20 MHz (durata ciclului instrucțiunii de 200 ns);
- memoria:
  - maximum 8k cuvinte de 14 biți de memorie FLASH;
  - maximum 368 octeți de memorie de date - Data Memory - (RAM);
  - maximum 256 octeți de memorie de date EEPROM;
- microcontrolerul este compatibil pin la pin cu PIC16C73B/74B/76/77
- posibilitatea tratării întreruperilor de la 14 surse de întrerupere;
- stivă hardware cu opt nivele;
- moduri de adresare directă, indirectă şi relativă;

- Power-on Reset (POR) – reset la aplicarea tensiunii de alimentare;
- Power-up Timer (PWRT) – temporizare la aplicarea tensiunii de alimentare, şi Oscillator Start-up Timer (OST) – stabilizarea oscilatorului la pornire;
- Wtchdog Timer (WDT) cu propriul oscilator RC on-chip pentru funcționare sigură;
- mecanism de protecție a codului programabil;
- mod SLEEP pentru economisirea energiei;
- opțiuni selectabile pentru oscilator;
- tehnologie CMOS FLASH/EEPROM de consum redus şi de viteză ridicată;
- circuitul este în întregime de tip static;
- In-Circuit Serial Programming (ICSP) - programarea serială directă a circuituui – prin intermediul a doi pini;
- posibilitate de programare serială cu o singură tensiune de 5V;
- In-Circuit Debugging – depanare directă la circuit – prin intermediul a doi pini;
- acces scriere/citire a procesorului la memoria de program;
- domeniul tensiunilor de alimentare: 2,0V la 5,5V;
- curent maxim absorbit: 25mA;

Conexiunile externe



Microcontrolerele PIC16F876/873 au 28 de pini iar microcontrolerele PIC16F877/874 au 40 de pini. Portul paralel slave nu este implementat la dispozitivele cu 28 de pini.

Descrierea pinilor pentru PIC16F874 şi PIC16F877

Numele pinului	DIP Nr. pin	PLCC Nr. pin	tip I/O/P	Tip buffer	Descriere
OSC1/CLKIN	13	14	I	ST/CMOS(4)	Intrarea oscilatorului cu cristal/sursă externă de ceas
OSC2/CLKOUT	14	15	O	-	Ieşirea oscilatorului cu cristal. Conectarea la cristal sau la rezonator în modul oscilator cu cristal. În modul RC, pinul OSC2 furnizează CLKOUT care are 1/4 din frecvența la OSC1 şi indică durata unui ciclu de instrucțiune.
~~MCLR~~/VPP	1	2	I/P	ST	Intrarea resetului principal – Master Clear (Reset) sau intrarea tensiunii de programare. Acest pin este activ în zero atunci când se aplică RESET.
					PORTA este un port I/O bidirectional
RA0/AN0	2	3	I/O	TTL	RA0 sau intrarea pentru semnal analogic 0.
RA1/AN1	3	4	I/O	TTL	RA1 sau intrarea pentru semnal analogic 1.
RA2/AN2/VREF-	4	5	I/O	TTL	RA2 sau intrarea pentru semnal analogic 2 sau referința negativă a semnalului analogic.
RA3/AN3/VREF+	5	6	I/O	TTL	RA3 sau intrarea pentru semnal analogic 3 sau referința pozitivă a semnalului analogic.
RA4/T0CKI	6	7	I/O	ST	RA4 sau intrarea de ceas a Timer0. Ieşirea este de tip open drain.
RA5/~~SS~~/AN4	7	8	I/O	TTL	RA5 sau intrarea pentru semnal analogic 4 sau selecția slave pentru portul serial sincron.
					PORTB este un port I/O bidirecțional. La PORTB poat fi conectate prin program rezistențe interne la toate intrările.
RB0/INT	33	36	I/O	TTL/ST(1)	RB0 sau pin pentru întreruperea externă
RB1	34	37	I/O	TTL	RB1
RB2	35	38	I/O	TTL	RB2
RB3/PGM	36	39	I/O	TTL	RB3 sau intrare de programare de tensiune scăzută
RB4	37	41	I/O	TTL	Pin cu generarea unei întreruperi la schimbare

RB5	38	42	I/O	TTL	Pin cu generarea unei întreruperi la schimbare sau pin In-Circuit Debugger. Ceasul programării seriale.
RB6/PGC	39	43	I/O	TTL/ST(2)	
RB7/PGD	40	44	I/O	TTL/ST(2)	Pin cu generarea unei întreruperi la schimbare sau pin In-Circuit Debugger. Datele programării seriale.
					PORTC este un port I/O bidirecțional
RC0/T1OSO/T1CKI	15	16	I/O	ST	RC0 sau ieşirea oscilatorului Timer 1 sau intrerea de ceas Timer 1
RC1/T1OSI/CCP2	16	18	I/O	ST	RC1 sau intrarea oscilatorului Timer 1 sau intrerea Capture2/Ieşirea Comare2/ieşirea PWM2
RC2/CCP1	17	19	I/O	ST	RC2 sau intrerea Capture1/Ieşirea Comare1/ieşirea PWM1
RC3/SCK/SCL	18	20	I/O	ST	RC3 sau intrarea/ieşirea ceasului serial sincron pentru modurile SPI şi I^2C
RC4/SDI/SDA	23	25	I/O	ST	RC4 sau intrare de date SPI (în modul SPI) sau data I/O (în modul I^2C)
RC5/SDO	24	26	I/O	ST	RC5 sau ieşire de date SPI (în modul SPI)
RC6/TX/CK	25	27	I/O	ST	RC6 sau ieşirea de date a transmițătorului USART sau ceasul sincron
RC7/RX/DT	26	29	I/O	ST	RC7 sau intrerea de date a tramsmițătorului USART sau datele sincrone
					PORTD este un port bidirecțional I/O sau un port paralel slave atcnci când se interfațează cu bus-ul unui microprocesor
RD0/PSP0	19	21	I/O	ST/TTL(3)	
RD1/PSP1	20	22	I/O	ST/TTL(3)	
RD2/PSP2	21	23	I/O	ST/TTL(3)	
RD3/PSP3	22	24	I/O	ST/TTL(3)	
RD4/PSP4	27	30	I/O	ST/TTL(3)	
RD5/PSP5	28	31	I/O	ST/TTL(3)	
RD6/PSP6	29	32	I/O	ST/TTL(3)	
RD7/PSP7	30	33	I/O	ST/TTL(3)	
					PORTE este un port bidirecțional I/O
RE0/~~RD~~/AN5	8	9	I/O	ST/TTL(3)	RE0 sau comanda de citire pentru portul paralel slave sau intrarea analogică 5

RE1/~~WR~~/AN6 RE2/~~CS~~/AN7	9 10	10 11	I/O I/O	ST/TTL(3) ST/TTL(3)	RE1 sau comanda de scriere pentru portul paralel slave sau intrarea analogică 6 RE2 sau comanda de selacție pentru portul paralel slave sau intrarea analogică 7
VSS	12,31	13,34	P	-	Masa (referința de tensiune) pentru pinii digitali I/O
VDD	11,32	12,35	P	-	Tensiunea de alimentare pozitivă prntru pinii digitali I/O
NC	-	1,17, 28,40		-	Aceşti pini nu sunt conectați intern. Aceşti pini trebuie să rămână neconectați

Legendă: I = intrare O = ieşire I/O = intrare/ieşire P = alimentare
- = nefolosit TTL = intrare TTL ST = intrare trigger Schmitt
~~SEMNAL~~ = semnalul SEMNAL negat

Notă:
1: acest buffer este o intrare trigger Schmitt când este configurat ca întrerupere externă;
2: acest buffer este o intrare trigger Schmitt când este folosit în modul de programare serială;
3: acest buffer este o intrare trigger Schmitt când este configurat I/O de uz general şi ca intrare TTL când este folosit în modul port paralel slave (pentru interfațarea cu magistrala unui microprocesor);
4: aceast buffer este o intrare trigger Schmitt când este configurat în modul oscilator RC şi intrare CMOS în celelalte cazuri.

Caracteristici principale	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Frecvența de lucru	DC-20 MHz	DC-20 MHz	DC-20 MHz	DC-20 MHz
RESET (Întârzieri)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Memoria de program FLASH (cuvinte de 14 biți)	4k	4k	8k	8k
Memoria de date (octeți)	192	192	368	368
Memoria de date EEPROM (octeți)	128	128	256	256
Întreruperi	13	14	13	14
Porturi I/O	Porturile A,B,C	Porturile A,B,C,D,E	Porturile A,B,C	Porturile A,B,C,D,E
Timer-e	3	3	3	3
Module Captură/Comparare/PWM	2	2	2	2
Comunicație serială	MSSP,	MSSP,	MSSP,	MSSP,

	USART	USART	USART	USART
Comunicație paralelă	-	PSP	-	PSP
Modul de conversie analog-digitală pe 10 biți	5 canale	8 canale	5 canale	8 canale
Set de instrucțiuni	35 instrucțiuni	35 instrucțiuni	35 instrucțiuni	35 instrucțiuni

Organizarea memoriei

Microcontrolerul are 3 blocuri de memorie. Memoria de program şi memoria de date au magistrale separate şi deci se pote realiza accesul simultan la date şi la program.

Contorul de program are 13 biți ce poate adresa un spațiu de memorie program de 8K x 14 biți. Accesarea locațiilor în afara spațiului fizic al memoriei implementate va produce o adresare wraparound.

Vectorul RESET este 0000h iar vectorul de întrerupere este 0004h.

Memoria este paginată iar paginile sunt:

- pagina 0 la adresele 0005h la 07FFh inclusiv;
- pagina 1 la adresele 0800h la 0FFFh inclusiv;
- pagina 2 la adresele 1000h la 17FFh inclusiv;
- pagina 3 la adresele 1800h la 1FFFh inclusiv.

Organizarea memoriei de date

Memoria de date este împărțită în 4 bank-uri ce conțin registrele de uz general (General Purpose Registers) şi registrele funcțiilor speciale (Special Function Registers).

Selecția bank-urilor se face cu ajutorul biților RP1 (STATUS<6>) şi RP0 (STATUS<5>).

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Fiecare bank din memoria statică are 128 de octeți (7Fh). Locațiile la adresele mici sunt rezervate regiătrilor funcțiilor speciale (SFR) iar sub acestea se găsesc regiştrii de uy general (GPR). Fiecare bank are proprii regiştrii SFR dar anumiți regiştrii SFR dintr-un bank se pot găsi şi într-un alt bank pentru reducerea dimensiunii codului şi pentru acces mai rapid la aceşti regiştrii.

Regiştrii de uz general pot fi accesați atât în mod direct cât şi indirect prin intermediul regiştrilor de selecție: File Select Register (FSR).

Harta regiştrilor microcontrolerelor PIC16F877/876

Bank 0		Bank 1		Bank 2		Bank 3	
Adresa de mem.	Registrul din memorie	Adresa de mem.	Registrul din memorie	Adresa de mem.	Registrul din memorie	Adresa de mem.	Registrul din memorie
00h	adresare indirectă(*)	80h	adresare indirectă(*)	100h	adresare indirectă(*)	180h	adresare indirectă(*)
01h	TMR0	81h	OPTION_REG	101h	TMR0	181h	OPTION_REG
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PORTA	85h	TRISA	105h		185h	
06h	PORTB	86h	TRISB	106h	PORTB	186h	TRISB
07h	PORTC	87h	TRISC	107h		187h	
08h	PORTD(1)	88h	TRISD(1)	108h		188h	
09h	PORTE(1)	89h	TRISE(1)	109h		189h	
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh	INTCON
0Ch	PIR1	8Ch	PIE1	10Ch	EEDATA	18Ch	EECON1
0Dh	PIR2	8Dh	PIE2	10Dh	EEADR	18Dh	EECON2
0Eh	TMR1L	8Eh	PCON	10Eh	EEDATH	18Eh	rezervat(2)
0Fh	TMR1H	8Fh		10Fh	EEADRH	18Fh	rezervat(2)
10h	T1CON	90h		110h	Regiştrii de uz general 16 octeți	190h	Regiştrii de uz general 16 octeți
11h	TMR2	91h	SSPCON2	111h		191h	
12h	T2CON	92h	PR2	112h		192h	
13h	SSPBUF	93h	SSPADD	113h		193h	
14h	SSPCON	94h	SSPSTAT	114h		194h	
15h	CCPR1L	95h		115h		195h	
16h	CCPR1H	96h		116h		196h	
17h	CCP1CON	97h		117h		197h	
18h	RCSTA	98h	TXSTA	118h		198h	
19h	TXREG	99h	SPBRG	119h		199h	
1Ah	RCREG	9Ah		11Ah		19Ah	
1Bh	CCPR2L	9Bh		11Bh		19Bh	
1Ch	CCPR2H	9Ch		11Ch		19Ch	
1Dh	CCP2CON	9Dh		11Dh		19Dh	
1Eh	ADRESH	9Eh	ADRESL	11Eh		19Eh	
1Fh	ADCON0	9Fh	ADCON1	11Fh		19Fh	
20h 7Fh	Regiştrii de uz general 96 octeți	A0h EFh	Regiştrii de uz general 80 octeți	120h 16Fh	Regiştrii de uz general 80 octeți	1A0h 1EFh	Regiştrii de uz general 80 octeți
		F0h FFh	acces 70h-7Fh	170h 17Fh	acces 70h-7Fh	1F0h 1FFh	acces 70h-7Fh

Locație de memorie neimplementată, la citire se citeşte zero.

* Nu este un registru fizic.

Notă: 1: aceste registre nu sunt implementate la PIC16F876.
2: aceste registre sunt rezervate, ele nu trebuie scrise.

Biții de configurare (directiva __CONFIG)

__CONFIG Directive Symbols (From Microchip Header Files)

Feature	SYMBOLS
Oscillators	_RC_OSC
	_EXTRC_OSC
	_EXTRC_OSC_CLKOUT
	_EXTRC_OSC_NOCLKOUT
	_INTRC_OSC
	_INTRC_OSC_CLKOUT
	_INTRC_OSC_NOCLKOUT
	_LP_OSC
	_XT_OSC
	_HS_OSC
Watch Dog Timer	_WDT_ON
	_WDT_OFF
Power-up Timer	_PWRTE_ON
	_PWRTE_OFF
Brown-out Reset	_BODEN_ON
	_BODEN_OFF
Master Clear Enable	_MCLRE_ON
	_MCLRE_OFF
Code Protect	_CP_ALL
	_CP_ON
	_CP_75
	_CP_50
	_CP_OFF
Code Protect Data EEPROM	_DP_ON
	_DP_OFF
Code Protect Calibration Space	_CPC_ON
	_CPC_OFF

Note 1: Not all configuration bit symbols may be available on any one device. Please refer to the Mlcrochip include file of that device for available symbols.

```
;==========================================================================
;
;       Configuration Bits
;
;==========================================================================

_CP_ALL                 EQU     H'1FFF'
_CP_OFF                 EQU     H'3FFF'
_DEBUG_OFF              EQU     H'3FFF'
_DEBUG_ON               EQU     H'37FF'
_WRT_OFF                EQU     H'3FFF'     ; No prog memmory write
protection
```

```
_WRT_256                EQU   H'3DFF'     ; First 256 prog memmory write
protected
_WRT_1FOURTH            EQU   H'3BFF'     ; First quarter prog memmory write
protected
_WRT_HALF               EQU   H'39FF'     ; First half memmory write
protected
_CPD_OFF                EQU   H'3FFF'
_CPD_ON                 EQU   H'3EFF'
_LVP_ON                 EQU   H'3FFF'
_LVP_OFF                EQU   H'3F7F'
_BODEN_ON               EQU   H'3FFF'
_BODEN_OFF              EQU   H'3FBF'
_PWRTE_OFF              EQU   H'3FFF'
_PWRTE_ON               EQU   H'3FF7'
_WDT_ON                 EQU   H'3FFF'
_WDT_OFF                EQU   H'3FFB'
_RC_OSC                 EQU   H'3FFF'
_HS_OSC                 EQU   H'3FFE'
_XT_OSC                 EQU   H'3FFD'
_LP_OSC                 EQU   H'3FFC'
```

Cuvântul de configurare (adresa 2007h)

Valoarea citită pentru biții de configurare neprogramați (șterși) este 3FFFh.

Un bit programat este citit ca zero iar cel neprogramat este citit ca unu. Aceşti biți se găsesc în spațiul de program şi locația de memorie de la adresa 2007h poate fi modificată numai pe timpul programării.

R/P-1	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-0	U-0	R/P-1	R/P-1	R/P-1
CP	-	DEBUG	WRT1	WRT0	CPD	LVP	BORDEN	-	-	~~PWRTEN~~	WDTEN	F0SC1	F0SC0
Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

bit 13 CP: bit de protecție a codului din memoria FLASH

1 = protecția codului off
0 = codul din memoria de program este protejat – odată protejată, memoria FLASH nu mai poate fi scrisă.

bit 12 Neimplementat. Citit ca unu.

bit 11 DEBUG: bit pentru depanare In-Circuit

1 = depanator In Circuit dezactivat. RB6 şi RB7 sunt pini de uz general I/O
0 = depanator In Circuit activat. RB6 şi RB7 sunt atribuiți depanatorului

bit 10-9 WRT1:WRT0 biți de validare a scrierii memoriei de program FLASH

11 = nu este activată protecția la scriere; toată memoria de program poate fi scrisă sub controlul EECON
10 = 0000h la 00FFh este protejat la scriere; 0100h la 1FFFh poate fi scrisă sub controlul EECON
01 = 0000h la 07FFh este protejat la scriere; 0800h la 1FFFh poate fi scrisă sub controlul EECON
00 = 0000h la 0FFFh este protejat la scriere; 1000h la 1FFFh poate fi scrisă sub controlul EECON

bit 8 CPD: bit de protecție a memoriei de date EEPROM

1 = memoria EEPROM nu este protejata
0 = memoria EEPROM este protejată

bit 7 LVP: bit de validare a tensiunii scăzute la programarea serială in-circuit

1 = pinii RB3/PGM au funcția PGM; programarea cu tensiune scăzută este validată;
0 = RB3 este I/O digitală, trebuie utilizată HV la ~~MCLR~~ pentru programare

bit 6 BORDEN: bit de validare a Brown-out Reset
1 = BOR activat
0 = BOR dezactivat

bit 5-4 Neimplementat. Citit ca unu.

bit 3 ~~PWRTEN~~ bit de validare a Power-up Timer

1 = PWRT dezactivat
0 = PWRT activat

bit 2 WDTEN: bit de validare a Watchdog Timer

1 = WDT activat
0 = WDT dezactivat

bit 1-0 FOSC1:FOSC0: biții de selecție a oscilatorului

11 = oscilator RC
10 = oscilator HS
01 = oscilator XT

00 = oscilator LP

Regiştrii cu funcţii speciale

Adr.	Nume	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valoarea în: POR, BOR
Bank 0										
0h(3)	INDF	Adresând această locaţie de memorie se foloseşte conţinutul FSR pentru a adresa memoria de date (nu este un registru fizic)								0000 0000
01h	TMR0	Registrul Timer0								xxxx xxxx
02h(3)	PCL	Contorul de program (PC) cel mai puţin semnificativ octet								0000 0000
03h(3)	STATUS	IRP	RP1	RP0	~~TO~~	~~PD~~	Z	DC	C	0001 1xxx
04h(3)	FSR	Pointer pentru adresarea indirectă a memoriei de date								xxxx xxxx
05h	PORTA	-	-	Memorare date PORTA la scriere, pini PORTA la citire						--0x 0000
06h	PORTB	Memorare date PORTB la scriere, pini PORTB la citire								xxxx xxxx
07h	PORTC	Memorare date PORTC la scriere, pini PORTC la citire								xxxx xxxx
08h(4)	PORTD	Memorare date PORTD la scriere, pini PORTD la citire								xxxx xxxx
09h(4)	PORTE	-	-	-	-	-	RE2	RE1	RE0	---- -xxx
0Ah(1,3)	PCLATH	-	-	-	Buffer de scriere a celor mai semnificativi 5 biţi ai contorului de program (PC)					---0 0000
0Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000
0Dh	PIR2	-	CMIF	-	EEIF	BCLIF	-	-	CCP2IE	-0-0 0—0
0Eh	TMR1L	Registrul de stocare al celui mai puţin semnificativ octet al registrului de 16 biţi al TMR1								xxxx xxxx
0Fh	TMR1H	Registrul de stocare al celui mai semnificativ octet al registrului de 16 biţi al TMR1								xxxx xxxx
10h	T1CON	-	-	T1CKPS1	T1CKPS0	T1OSCEN	~~T1SYNC~~	TMR1CS	TMR1ON	--00 0000
11h	TMR2	Registrul Timer2								1111 1111
12h	T2CON	-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS1	TMR2ON	T2CKPS1	T2CKPS0	-000 0000
13h	SSPBUF	Registrul de transmisie/buffer de recepţie a portului serial sincron								xxxx xxxx
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000
15h	CCPR1L	Registrul 1 (LSB) Captură/Comparare/PWM								0000 0000
16h	CCPR1H	Registrul 1 (MSB) Captură/Comparare/PWM								-
17h	CCP1CON	-	-	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M2	CCP1M0	--00 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x
19h	TXREG	Registrul datelor de transmis al USART								0000 0000
1Ah	RCREG	Registrul datelor recepţionate al USART								0000 0000
1Bh	CCPR2L	Registrul 2 (LSB) Captură/Comparare/PWM								xxxx xxxx
1Ch	CCPR2H	Registrul 2 (MSB) Captură/Comparare/PWM								xxxx xxxx
1Dh	CCP2CON	-	-	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000
1Eh	ADRESH	Registrul rezultat al octetului cel mai semnificativ al conversiei A/D								xxxx xxxx
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/~~DONE~~	-	ADON	0000 00-0

Legendă: x = necunoscut, u = neschimbat, q = valoarea depinde de situaţie, - = neimplementat – se citeşte zero, r = rezervat

Notă: 1. Cel mai semnificativ octet al contorului de program nu este accesibil direct. Registrul PCLATH conține biții PC<12:8> care vor fi transferați în octetul cel mai semnificativ al contorului de program.
2. Biții PSPIE şi PSPIF sunt rezervați la PIC16F873A/876A şi trebuie menținuți şterşi.
3. Aceşti regiştrii pot fi adresați din orice bank.
4. PORTD, PORTE, TRISD şi TRISE nu sunt implementați la PIC16F873A/876A şi la citire sunt zero.
5. Bitul 4 al EEADRH este implementet numai la PIC16F876A/877A.

Adr.	Nume	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valoarea în: POR, BOR
Bank 1										
80h(3)	INDF	Adresând această locație de memorie se folosește conținutul FSR pentru a adresa memoria de date (nu este un registru fizic)								0000 0000
81h	OPTION_REG	~~RBPU~~	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111
82h(3)	PCL	Contorul de program (PC) cel mai puțin semnificativ octet								0000 0000
83h(3)	STATUS	IRP	RP1	RP0	~~TO~~	~~PD~~	Z	DC	C	0001 1xxx
84h(3)	FSR	Pointer pentru adresarea indirectă a memoriei de date								xxxx xxxx
85h	TRISA	-	-	Registru direcție PORTA						--11 1111
86h	TRISB	Registru direcție PORTB								1111 1111
87h	TRISC	Registru direcție PORTC								1111 1111
88h(4)	TRISD	Registru direcție PORTD								1111 1111
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	-	Biți direcție PORTE			0000 -111
8Ah(1,3)	PCLATH	-	-	-	Buffer de scriere a celor mai semnificativi 5 biți ai contorului de program (PC)					---0 0000
8Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000
8Dh	PIE2	-	CMIE	-	EEIE	BCLIE	-	-	CCP2IE	-0-0 0—0
8Eh	PCON	-	-	-	-	-	-	~~POR~~	~~BOR~~	---- --qq
8Fh	-	Neimplementat								-
90h	-	Neimplementat								-
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000
92h	PR2	Registru de perioada al Timer2								1111 1111
93h	SSPADD	Registrul de adresă al portului serial sincron (mod I^2C)								0000 0000
94h	SSPSTAT	SMP	CKE	D/~~A~~	P	S	R/~~W~~	UA	BF	0000 0000
95h	-	Neimplementat								-
96h	-	Neimplementat								-
97h	-	Neimplementat								-
98h	TXSTA	CSRC	TX9	TXEN	SZNC	-	BRGH	TRMT	TX9D	0000 -010
99h	SPBRG	Registrul generator Baud Rate								0000 0000
9Ah	-	Neimplementat								-
9Bh	-	Neimplementat								-
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111
9Dh	CVRCON	CVREN	CVROE	CVRR	-	CVR3	CVR2	CVR1	CVR0	000- 0000
9Eh	ADRESL	Registrul octetului cel mai puțin semnificativ al conversiei A/D								xxxx xxxx
9Fh	ADCON1	ADFM	ADCS2	-	-	PCFG3	PCFG2	PCFG1	PCHG1	0--- 0000

Legendă: x = necunoscut, u = neschimbat, q = valoarea depinde de situație, - =

neimplementat – se citeşte zero, r = rezervat

Notă: 1. Cel mai semnificativ octet al contorului de program nu este accesibil direct. Registrul PCLATH conține biții PC<12:8> care vor fi transferați în octetul cel mai semnificativ al contorului de program.
2. Biții PSPIE şi PSPIF sunt rezervați la PIC16F873A/876A şi trebuie menținuți şterşi.
3. Aceşti regiştrii pot fi adresați din orice bank.
4. PORTD, PORTE, TRISD şi TRISE nu sunt implementați la PIC16F873A/876A şi la citire sunt zero.
5. Bitul 4 al EEADRH este implementet numai la PIC16F876A/877A.

Adr.	Nume	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valoarea în: POR, BOR
Bank 2										
100h(3)	INDF	Adresând această locație de memorie se folosește conținutul FSR pentru a adresa memoria de date (nu este un registru fizic)								0000 0000
101h	TMR0	Registrul Timer0								xxxx xxxx
102h(3)	PCL	Contorul de program (PC) cel mai puțin semnificativ octet								0000 0000
103h(3)	STATUS	IRP	RP1	RP0	~~TO~~	~~PD~~	Z	DC	C	0001 1xxx
104h(3)	FSR	Pointer pentru adresarea indirectă a memoriei de date								xxxx xxxx
105h	-	Neimplementat								
106h	PORTB	Memorare date PORTB la scriere, pini PORTB la citire								xxxx xxxx
107h	-	Neimplementat								-
108h	-	Neimplementat								-
109h	-	Neimplementat								-
10Ah(1,3)	PCLATH	-	-	-	Buffer de scriere a celor mai semnificativi 5 biți ai contorului de program (PC)					---0 0000
10Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x
10Ch	EEDATA	Octetul cel mai puțin semnificativ (L) al registrului de date al EEPROM								xxxx xxxx
10Dh	EEADR	Octetul cel mai puțin semnificativ (L) al registrului de adresă al EEPROM								xxxx xxxx
10Eh	EEDATH	-	-	Octetul cel mai semnificativ (H) al registrului de date al EEPROM						--xx xxxx
10Fh	EEADRH	-	-	-	-(5)	Octetul cel mai semnificativ (H) al registrului de adresă al EEPROM				---- xxxx
Bank 3										
180h(3)	INDF	Adresând această locație de memorie se folosește conținutul FSR pentru a adresa memoria de date (nu este un registru fizic)								0000 0000
181h	OPTION_REG	~~RBPU~~	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111
182h(3)	PCL	Contorul de program (PC) cel mai puțin semnificativ octet								0000 0000
183h(3)	STATUS	IRP	RP1	RP0	~~TO~~	~~PD~~	Z	DC	C	0001 1xxx
184h(3)	FSR	Pointer pentru adresarea indirectă a memoriei de date								xxxx xxxx
185h	-	Neimplementat								
186h	TRISB	Registru de direcție a datelor PORTB								xxxx xxxx
187h	-	Neimplementat								-
188h	-	Neimplementat								-
189h	-	Neimplementat								-
18Ah(1,3)	PCLATH	-	-	-	Buffer de scriere a celor mai semnificativi 5 biți ai contorului de program (PC)					---0 0000
18Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x
18Ch	EECON1	EEPGD	-	-	-	WRERR	WREN	WR	RD	x--- x000

18Dh	EECON2	Registrul de control 2 (nu este un registru fizic) EEPROM	---- ----
18Eh	-	Rezervat, trebuie menţinut şters	0000 0000
18Fh	-	Rezervat, trebuie menţinut şters	0000 0000

Legendă: x = necunoscut, u = neschimbat, q = valoarea depinde de situaţie, - = neimplementat – se citeşte zero, r = rezervat

Notă:
1. Cel mai semnificativ octet al contorului de program nu este accesibil direct. Registrul PCLATH conţine biţii PC<12:8> care vor fi transferaţi în octetul cel mai semnificativ al contorului de program.
2. Biţii PSPIE şi PSPIF sunt rezervaţi la PIC16F873A/876A şi trebuie menţinuţi şterşi.
3. Aceşti regiştrii pot fi adresaţi din orice bank.
4. PORTD, PORTE, TRISD şi TRISE nu sunt implementaţi la PIC16F873A/876A şi la citire sunt zero.
5. Bitul 4 al EEADRH este implementet numai la PIC16F876A/877A.

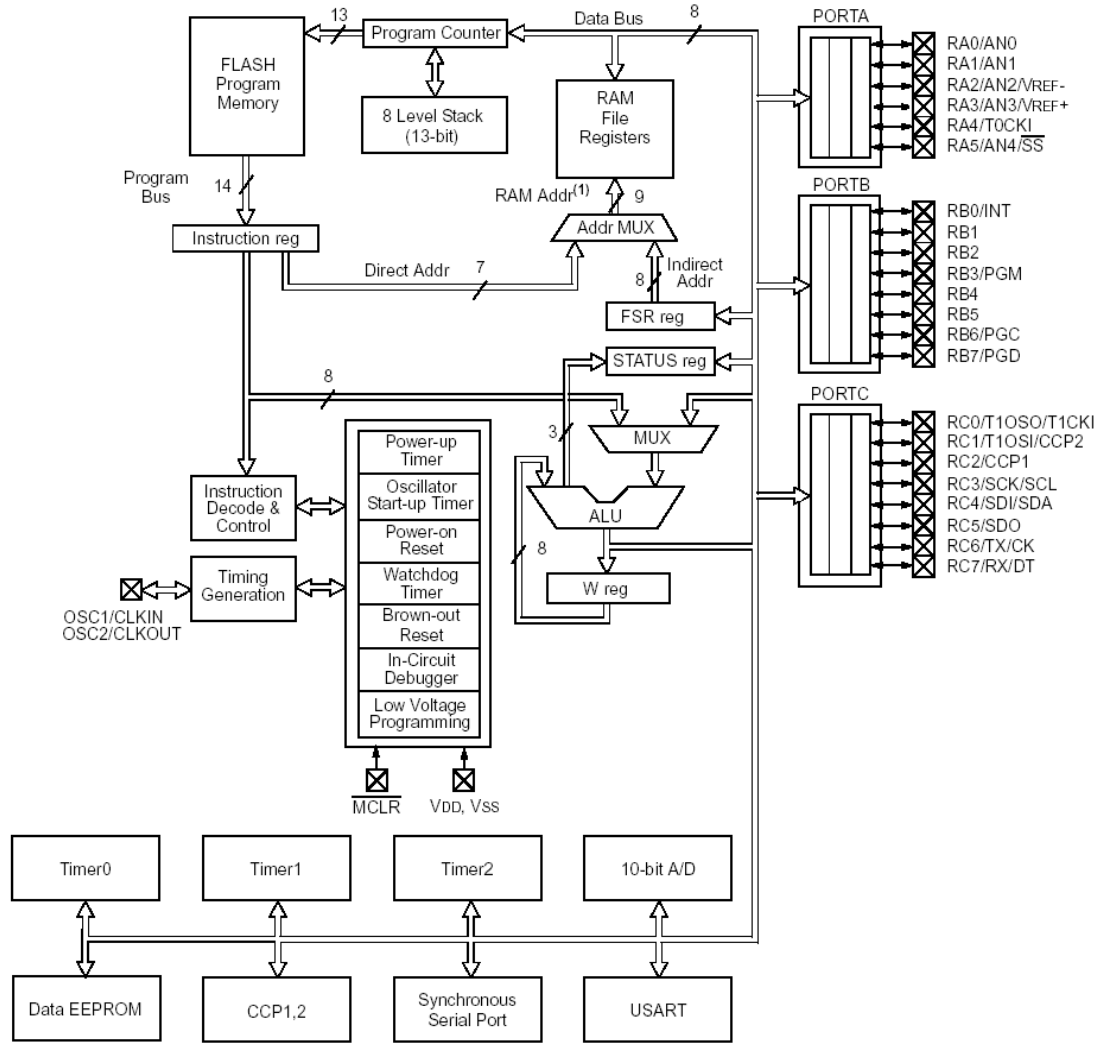


Figura 4.16. Schema bloc a microcontrolerului PIC16F877.

CAPITOLUL 5

MEMORIA

5.1. Prezentare generală

Memoria unui calculator are rolul de a stoca temporar sau permanent date sau programe. Durata de stocare a datelor se referă la timpul cât memoria este alimentată la tensiunea electrică. O memorie poate stoca permanent date dacă informația memorată nu se pierde la întreruperea tensiunii de alimentare a memoriei. În caz contrar, atunci când memoria stochează informația numai pe durata de timp cât este alimentată cu energie electrică, se spune că informația este memorată temporar. Este momentul sa precizăm faptul că un calculator numeric nu prelucrează decât numere. Din acest motiv împărțirea informației în date şi instrucțiuni are caracter pur convențional. Astfel, într-o locație de memorie se poate afla numărul ($10110110_B = 182_Z$) care pentru unitatea centrală poate însemna fie o valoare numerică egală cu 128 fie codul unei instrucțiuni de tipul: 'aduna numarul a cu numarul b'. Unitatea centrală decide dacă numărul citit din memorie reprezintă o valoare numerică (dată) sau o comandă (instrucțiune program). Prin convenție, la pornire sau după inițializare (RESET) unitatea centrală consideră că valoarea citită din memorie reprezintă o comandă. Să presupunem ca acesta comanda este: 'aduna numarul a cu numarul b'. Asta înseamna ca urmatoarele două valoari citite din memorie se vor considera ca reprezentând valoarile numerelor a şi b. După executarea adunării unitatea centrală va citi o nouă valoare din memorie pe care o va considera în mod automat ca reprezentând codul numeric al unei instrucțiuni de program. Rămâne deci responsabilitatea programatorului ca în memorie să fie înscrisă o succesiune corectă de date care reprezintă numere sau coduri de instrucțiuni.

După modul în care sunt stocate datele în memorie se poate face o primă clasificare a acestora:

- memorii de tip ROM (Read Only Memory) care sunt memorii ce pot stoca permanent datele. Aşa cum le arată şi numele ele nu pot fi decât citite de către unitatea centrală, înscrierea lor fiind făcută prin procedee speciale, fie pe calculator, fie separat, pe un dispozitiv special numit inscriptor de memorii ROM;
- memorii de tip RAM (Random Access Memory) care sunt memorii ce pot stoca temporar datele. Caracteristic acestor memorii este faptul că ele pot fi scrise şi citite în mod curent de unitatea centrală.

Cele două tipuri de memorie sunt amândouă folosite în calculator în scopuri diferite. Memoria de tip ROM stochează de obicei programul executat de unitatea

centrală imediat după pornire sau la inițializare. Acest program se mumeşte program ***monitor*** sau program de ***boot***. Dacă în calculator s-ar folosi numai memorie de tip RAM problema pornirii calculatorului nu ar putea fi rezolvată. La punerea sub tensiune o memorie RAM are un conținut aleator care nu poate ajuta la pornirea calculatorului; cum unitatea centrală, după inițializare aşteaptă o comandă validă, rezultă că trebuie folosită o memorie care să nu-şi piardă conținutul atunci când tensiunea de alimentare este întreruptă.

Structura unei memorii este similară cu cea a unei matrici. Vom considera un exemplu în care în fiecare element al matricii se află un modul elementar de memorie care nu poate memora decât un bit adică o valoare numerică egală cu zero sau cu unu. Cea mai simplă structură de memorie este prezentată în figura 5.1. Localizarea celulei elementare de memorie care va fi citită sau scrisă la un moment dat se face cu ajutorul semnalelor de adresă iar valoarea citită sau valoarea de înscris în memorie este disponibilă pe respectiv este depusă pe magistrala de date. În acest fel se spune că selecția unei celule de memorie se face cu ajutorul semnalelor de pe magistrala de adrese iar valoarea citită din memorie sau înscrisă în memorie se face cu ajutorul semnalelor de pe magistrala de date.

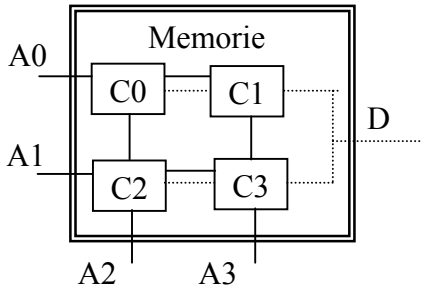


Fig. 5.1. Structura unei memorii

În figura 5.1, C0, C1, C2 şi C3 reprezintă celulele elementare de memorie, D magistrala de date (o singură linie) iar A0, A1, A2 şi A3 magistrala de adrese (patru linii). Traseul datelor a fost reprezentat cu linie punctată iar cel al adreselor cu linie continuă. Din cauză că la un moment dat numai una din celulele elementare de memorie este activată, pentru date este suficientă o singură linie. Considerăm că o celulă elementară de memorie este activată pentru citire sau scriere dacă ea se află la intersecția liniilor de adresă pe care se află simultan cifra unu. Funcționarea memoriei, în acest caz este dată în tabelul 5.1. Dacă pe magistrala de adrese se aplică, de exemplu, numărul 0101 (în succesiunea A3,A2,A1,A0) atunci pe magistrala de date se găseşte conținutul celulei C0 la citire sau poate fi înscrisă celula C0 cu valoarea plasată pe magistrala de date, la scriere.

TABELUL 5.1.

A3	A2	A1	A0	D
0	1	0	1	Conținutul celulei C0
1	0	0	1	Conținutul celulei C1
0	1	1	0	Conținutul celulei C2
1	0	1	0	Conținutul celulei C3

Din cauză că folosirea a patru linii de adresă este neeconomică în acest caz, în realitate la o astfel de memorie nu se află decât două linii de adresă din cauză că se pot obține patru combinații distincte numai cu două numere binare (00, 01, 10, 11). Translatarea de la cele patru combinații binare la cele patru adrese de selacție a celulelor de memorie se face în interiorul memoriei cu ajutorul unui circuit de decodificare.

Pentru a obține mai multe date simultan la ieşire memoriile se conectează în paralel aşa cum este reprezentat schematic în figura 5.2.

În structura din figura 5.2 dacă se aplică la intrare (pe liniile de adresă) combinația 0101 (în succesiunea A3,A2,A1,A0), atunci vor fi selectate simultan celulele

de memorie C00 şi C01, C00 fiind conectată la linia de date D0 iar C01 la D1. Toate celulele elementare de memorie selectate simultan, formează o celulă de memorie care permite manipularea simultana a mai multor biţi.

Din cele prezentate până acum rezultă că atât magistrala de adrese cât şi cea de date furnizează informaţii legate de dimensiunea (capacitatea memoriei).

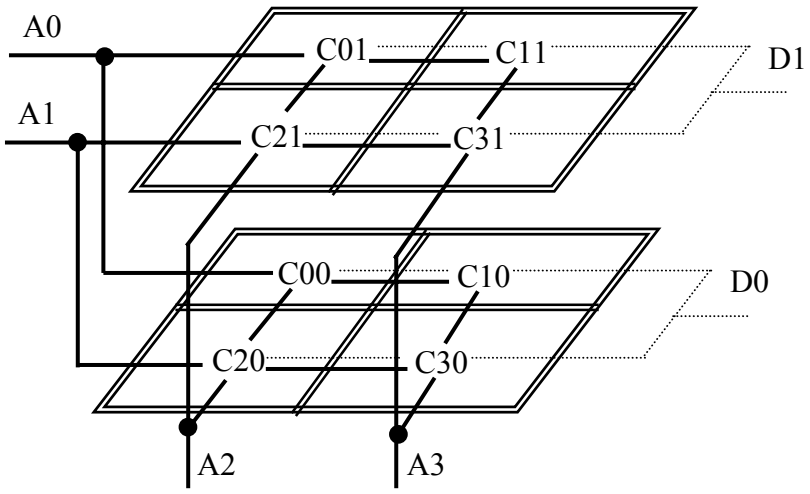


Fig.5.2. Structura unei memorii cu două planuri

Pentru a măsura capacitatea (dimensiunea) memoriei se foloseşte ca unitate de bază ***bitul*** (b) care reprezintă cantitatea elementară de informaţie. Spre exemplu, vom considera experimentul aruncării unei monezi. În urma acestui experiment înlăturăm o incertitudine de 50% pentru că înainte de aruncarea monezii existau numai două posibilităţi ale rezultatului (una din feţele monezii). Astfel, probabilitatea de obţinere a unui rezultat este de 1/2 din cauză că din cele două rezultate posibile s-a obţinut unul. Cantitatea de informaţie pe care o primim după efectuarea acestui experiment se exprimă cu relaţia:

$$I = - \log_2 P = - \log_2 ½ = 1 \text{ bit}$$

Dacă asociem celor două feţe ale monedei valorile binare 0 şi 1, cantitatea de informaţie stocată într-o celulă elementară de memorie (care poate memora una din valorile 1 sau 0) este de un bit. Bitul are ca multiplii ***kilobitul*** (kb), ***megabitul*** (Mb), ***gigabitul*** (Gb) şi ***terabitul*** (Tb) cu următoarele relaţii între ele:

- 1kb = 1024 biţi;
- 1Mb = 1024kb = 1 048 576 biţi;
- 1Gb = 1024Mb = 1 048 576 kb = 1 073 741 824 biţi;
- 1Tb = 1024Gb = 1 048 576 Mb = 1 073 741 824 kb = 1 099 511 627 776 biţi.

Pentru a ne forma o imagine asupra cantităţii de informaţie care poate fi stocată, putem spune că dacă un bit stochează informaţia corespunzătoare unui eveniment echiprobabil cu două posibilităţi de realizare aşa cum este experimentul aruncării unei monezi, într-un gibabit se poate stoca informaţia dintr-o enciclopedie iar într-un terabit informaţia dintr-o bibliotecă de 500 de cărţi.

Întorcându-ne la figura 5.2 putem spune că această memorie are o capacitate de 4x1bit din cauză că sunt patru celule elementare de memorie care pot stoca informaţie,

la un moment dat având acces la o singură celulă. Similar, memoria din figura 5.2 are o capacitate de 4x2 biți din cauză că, la un moment dat, avem acces la două celule elementare de memorie simultan.

Aşa cum s-a arătat în capitolul 4, unitățile centrale pot lucra cu 8, 16, 32, 64, ... biți de date simultan. Din acest motiv un grup de opt biți se numeşte ***octet*** (byte, 1 byte=8 bits). Daca se utilizează 16 biți simultan adică doi octeți, aceştia formează un ***cuvânt*** (word) iar două cuvinte formează un ***pointer***. În tabelul 5.2 sunt sintetizate modurile de notare a grupurilor de biți.

TABELUL 5.2.

1 octet (byte) = 8 biți (bites)
1 cuvânt (word) = 2 octeți (bytes) = 16 biți (bites)
1 pointer (poiner) = 2 cuvinte (words) = 4 octeți (bytes) = 32 biți (bits)

Memoriile calculatoarelor se construiesc în general cu celule de opt biți de date iar capacitățile acestora sunt exprimate în kiloocteți (kilobytes), notat cu ko sau kb, un ko fiind egal cu 1024 octeți. Capacitățile uzuale folosite pentru memoriile calculatoarelor numerice variază de la 1ko şi pot ajunge până la zeci sau sute de Mo (megaocteți).

Calculul capacității unei memorii poate fi făcută în felul următor: sa presupunem că avem o memorie de 5Mo; rezultă că această memorie va avea:

5 x 1 048 576 x 8 biți = 41 943 040 biți

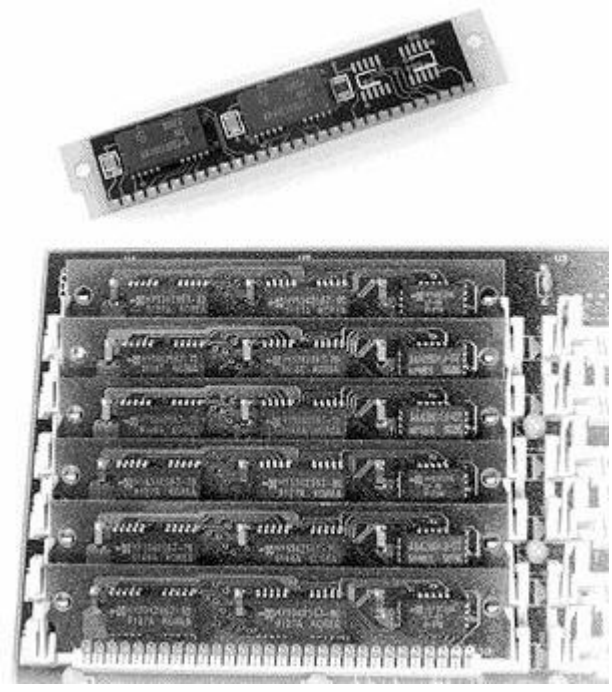


Fig. 5.3. Memorie SIMM şi modul de montare a acesteia

ceea ce înseamnă că o astfel de memorie va conține 41 943 040 celule elementare de memorie. În figura 5.3 se prezintă o memorie de 4 Mo şi modul de montare a acesteia.

În afară de capacitatea memoriei, o altă caracteristică importantă a acesteia este ***timpul de acces.*** Timpul de acces reprezintă timpul necesar memoriei de a răspunde unei comenzi. Cu alte cuvinte, la o comandă de citire, memoria nu depune instantaneu conținutul celulei de memorie (o celulă de memorie fiind formată din una sau mai multe celule elementare) pe magistrala de date ci după un anumit timp care depinde atât de tehnologia în care este realizată memoria cât şi de dimensiunea acesteia. La fel, la scrierea unei memorii, datele de înscris în memorie trebuie menținute

un anumit timp pe magistrala de date, pentru ca memoria să fie capabilă să le transfere în celula de memorare corespunzătoare. Este evident faptul că este de dorit ca timpul de acces al unei memorii să fie cât mai mic în aşa fel încât ea să poată răspunde suficient de repede solicitărilor unității centrale. În prezent memoriile utilizate în calculatoare au timpi de acces între 70ns şi 10ns. Din păcate cu cât timpul de acces al unei memorii este mai mic prețul acesteia este mai mare şi din acest motiv, în cele mai multe cazuri, memoria unui sistem este realizată dintr-o combinație de memorii rapide şi memorii lente. Astfel, memoria RAM într-un calculator este realizată pe cel puțin trei nivele. Pe primul nivel este memoria cea mai rapidă, care se află chiar în interiorul unității ceantrale. Această memorie se numeste memorie ***cache*** pe nivelul 1 (cache level 1). În această memorie se aduc porțiunile de program care sunt rulate la un moment dat, unitatea centrală lucrând practic numai cu memoria cache de pe nivelul 1. Atunci când este necesară execuția unei instrucțiuni care nu este în memoria cache de pe nivelul 1, un circuit de control al memoriei aduce porțiunea respectivă de program, din memoria externă mai lentă, în memoria cache de pe nivelul 1. Microprocesoarele actuale au o memorie cache internă între 128ko şi 512ko, iar la microprocesoarele foarte puternice destinate stațiilor de lucru, memoria cache poate ajunge până la 2Mo. În exteriorul unității centrale există un al doilea nivel al memoriei cache (cache level 2) cu acelaşi rol ca şi memoria cache level 1. Mecanismul memoriei cache este prezentat 1n figura 5.4. Controlerul de memorie are sarcina de a transfera datele de pe un nivel pe altul în aşa fel încât în memoria cache de pe nivelul 1 să se afle întotdeauna porțiunea de program necesar unității centrale.

Performanțele unui calculator vor fi cu atât mai bune cu cât dimensiunea memoriei cache este mai mare.

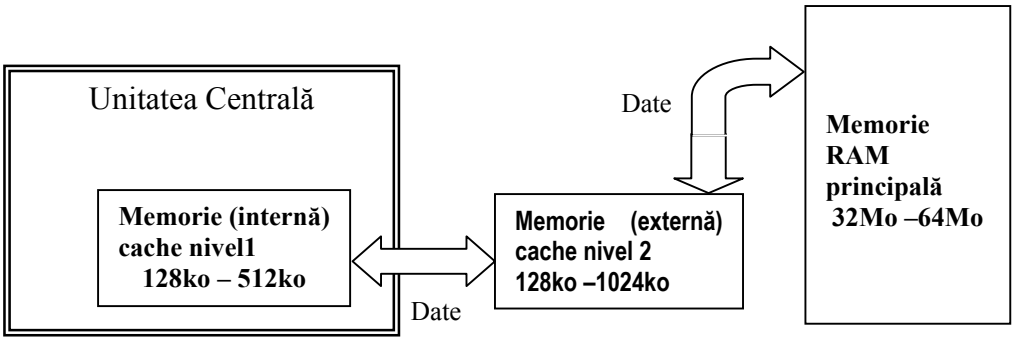


Fig. 5.4. Organizarea memoriei RAM la un calculator

În continuare se vor prezenta pe scurt principalele tipuri tehnologice de memorii ROM şi RAM existente în prezent şi caracteristicile acestora.

Memoriile ROM sunt realizate în următoarele variante:

- memorii ROM propriu-zise care nu pot fi înscrise decât o singură dată, înscrierea fiind făcută cel mai adesea la fabricant sau la utilizator cu ajutorul unor dispozitive de programare speciale, aceste memorii fiind denumite şi memorii PROM (Programmable Read-Only Memory);

- memorii EPROM (Erasable Programmable Read-Only Memory) care sunt memorii de tip ROM programabile. Ele pot fi sterse si înscrise (programate) de mai multe ori. Stergerea memoriei se face cu ajutorul unui flux de lumină ultravioletă, în acest scop capsula circuitului integrat fiind prevăzută cu un geam din sticlă de cuarț (vezi figura 4.1), iar înscrierea se face cu ajutorul unor dispozitive speciale prin metode electrice la tensiuni mai mari decât cele de funcționare normală;
- memorii de tip EEPROM (Electrically Erasable Programmable Read-Only Memory) care sunt memorii ROM programabile ce pot fi şterse şi reînscrise exclusiv prin mijloace electrice. Avantajul acestui tip de memorii este reprezentat de faptul că ele nu trebuie scoase din circuitul în care au fost montate pentru a fi şterse şi reînscrise. Acest tip de memorie necesită tensiuni mai mari decât cele de funcționare obişnuită, pentru ştergere şi programare;
- memorii de tip ***flash*** (flash memory), ultima generație de memorii de tip ROM care deşi păstrează informația şi după întreruperea tensiunii de alimentare, pot fi sterse si programate similar cu memoriile RAM. Singura particularitate este reprezentată de faptul că memoriile flash necesită algoritmi speciali de ştergere şi programare, această operațiune fiind făcută pe blocuri, în trei faze.

Memoriile RAM sunt realizate în următoarele variante:

- memorii RAM ***dinamice*** (DRAM - Dynamic Random Access Memory) care sunt memorii de tip RAM ce necesită reîmprospătarea periodică (la 20 ms) a conținutului, în caz contrar informația pierzându-se chiar dacă sunt alimentate. Reîmprospătarea se face simplu prin citirea periodică a mai multor blocuri simultan din memorie. Principalul avantaj al acestui tip de memorii este reprezentat de faptul că pot avea d densitate foarte mare a celulelor elementare, putând fi realizate memorii cu capacități foarte mari iar principalul dezavantaj este reprezentat de faptul că memoriile RAM dinamice sunt elativ lente (necesită timpi de acces mari). Din acest motiv memoriile RAM dinamice sunt folosite de regulă la memoria principală a calculatorului.
- memorii RAM ***statice*** (SRAM - Static Random Access Memory) care sunt memorii de tip RAM ce nu necesită reîmprospătare. Aceste memorii prezintă avantajul că au timpi de acces foarte mici în schimb au dezavantajul că nu pot fi realizate de capacități foarte mari. Memoriile RAM statice sunt utilizate de regulă la realizarea memoriilor cache.

Pentru cresterea performanțelor de viteză ale memoriilor RAM, în prezent sunt utilizate o serie de tehnici care au dus la apariția mai multor tipuri de memorii RAM dinamice:

- memorii de tip EDO DRAM (Extended Data Output Dynamic Random Access Memory) care sunt memorii de tip DRAM dar mai rapide decât memoriile DRAM obişnuite. Spre deosebire de memoriile DRAM obişnuite, o memorie EDO DRAM nu poate fi accesată decât pe blocuri de date. Pe durata de timp cât memoria EDO RAM trimite la unitatea centrală datele corespunzătoare unui bloc, poate căuta datele blocului următor. Datorită acestor mecanisme timpul de regăsire a datelor este mult redus;
- memorii de tip BEDO DRAM (Burst EDO DRAM). Memoria BEDO DRAM este mai rapidă decât memoria EDO DRAM. Acest tip de memorie se sincronizează cu viteza unității centrale pe durate scurte de timp (***burst***). Pe durata unei astfel de sincronizări o memorie BEDO DRAM poate prelucra patru adrese de memorie simultan. Aceste memorii se pot conecta la procesoare ce lucrează pe magistrale cu viteza de până la 66 MHz;
- memorii de tip SDRAM (Synchronous DRAM). Acest tip de memorii sunt capabile să se sincronizeze cu unitatea centrală şi funcționează la frecvențe de până la 200MHz (memorii **DDR SDRAM – double-data-rate synchronous dynamic random access memory)**.

În figura 5.5 este prezentat modul de realizare a circuitelor de memorie.

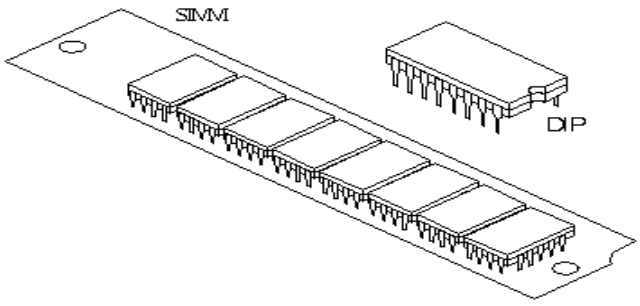


Fig. 5.5. Realizarea fizică a circuitelor de memorie

5.2. Aplicarea principiului "cache" în sistemele de calcul

5.2.1. Memoria Cache

Numele acestui tip de memorie provine de la termenul din limba franceză "ascuns": cache. Deşi este un cvânt de origine franceză s-a încetățenit pronunția acesuia

în limba engleză deoarece majoritatea termenilor din domeniul arhitecturii calculatoarelor provin din această limbă.

Memoria cache reprezintă mai degrabă un principiu: în sistemul de calcul se vor folosi două tipuri de memorie principală – o memorie lentă de dimensiuni mari şi o memorie rapidă de dimensiuni mai mici. Acest lucru se întâmplă din cauză că memoria rapidă este mult mai scumpă decât cea lentă şi atunci mecanismul memoriei cache se foloseşte în scopul cresterii performantelor sistemului în condiţii de preţ acceptabile.

Functionarea memoriei cache se bazează pe aducerea anumitor secvenţe de program, secvenţe ce sunt necesare la un moment dat, din memoria lentă în memoria rapidă (memoria cache) – figura 5.6.

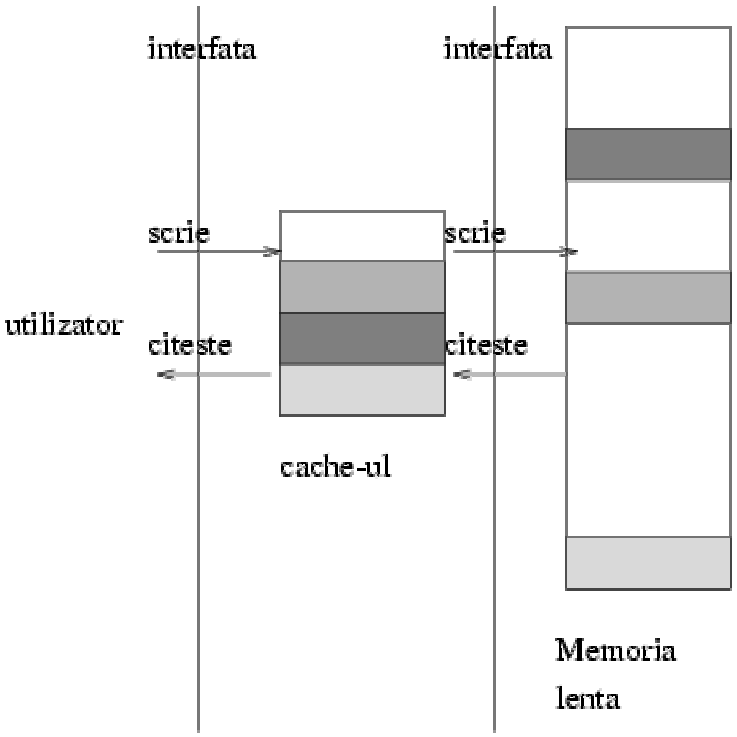


Figura 5.6. Principiul cache-ului

Funcţionarea acestei memrii se bazează pe principiul "localităţii". Există două feluri de localizarea a informaţiei: localizarea spaţială şi localizarea temporală:

- localitate *spaţială*: dacă la un moment dat sunt necesare nişte date, în curînd vor fi necesare probabil de date aflate în apropierea lor în memorie;
- localitatea *temporală*: cînd sunt folosite nişte date, foarte adesea acestea se vor folosi de mai multe ori.

Memoria cache foloseşte acelaşi fel de interfaţă ca şi memoria principală a sistemului. Acest lucru înseamnă că nu ne vom da seama de prezenţa sau lipsa memoriei cache decât prin viteza de calcul a sistemului.

Datorită faptului că eficienţa memoriei cache depinde în bună măsură de programul executat, pentru determinarea performanţelor memoriei cache se fac mai multe determinări cu ajutorul diferitelor tipuri de programe şi apoi se realizează o medie.

Eficienţa unui cache se măsoară în procentajul de situaţii când datele căutate se găsesc în cache H (*hit ratio*): din 100 de accese, cîte găsesc datele în cache? Opusul acestei valori este *miss ratio* M (rateuri). Procentajele astea se măsoară rulînd o mulţime de programe şi făcînd media. Avem desigur, dacă socotim 33% ca 1/3:

$$H = 1 - M$$

Dacă timpul de citire din cache este Tc (``hit time''), iar timpul pe care îl pierdem cînd ratăm este Tm (``miss time'') atunci putem măsura timpul mediu de acces la memoria cu cache cu următoarea formulă:

$$T = Tc \times H + Tm \times M$$

Trebuie observat faptul că timpul unei ratări (Tm) nu este neapărat egal cu timpul de citire din memoria lentă (Tl), pentru că în cazul unei ratări, întîi trebuie să ne dăm seama dacă datele sunt în cache, iar dacă nu sunt să accesăm memoria lentă. Cache-ul va fi eficient dacă T < Tl.

Valoarea mărimii H depinde de mărimea cache-ului: pentru un cache de mărimea memoriei lente (caz limită), toate datele pot fi ţinute în memoria rapidă, şi vom avea H=1. Pentru un cache de mărime 0, H=0, pentru că niciodată datele nu se găsesc în el. Relaţia între mărimea cache-ului, a memoriei lente şi H nu este o linie dreaptă, ci creşte repede la început (figura 5.7). Din cauza asta un cache relativ mic ca mărime are o importanţă mare ca eficienţă.

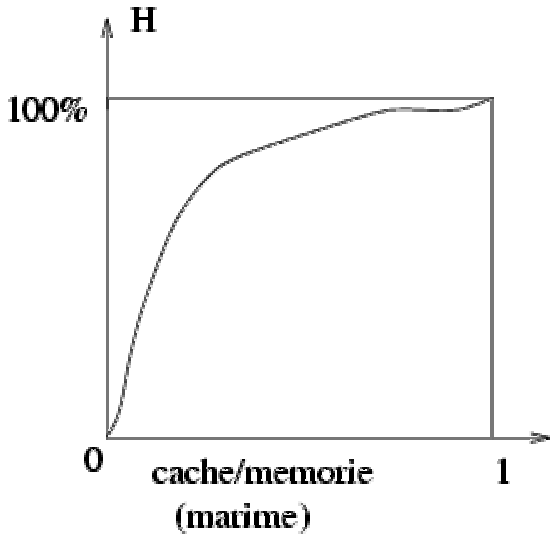


Figura 5.7. Performanţa cache-ului

Eficiența depinde şi de raportul dintre Tc şi Tm; în anumite cazuri Tm este de ordinul a 10000 x Tc, deci chiar un H mic poate să însemne mult.

Pe lîngă dimensiuni şi timpi de acces, există o mulțime de detalii prin care cache-urile diferă între ele, datorate faptului că mediile de stocare a informației nu se comportă chiar la fel. Iată unele dintre posibilele diferențe:

1) Mărimea blocului de date: cîteodată este mai economic să se transporte mai multe date deodată din memoria lentă; cache-urile aduc atunci mai mult decît li se cere (un calup) şi păstrează totul, în ideea (sugerată de principiul de localitate spațială) că şi vecinii obiectului accesat vor fi căutați în curînd. Unitatea de transfer între cache şi memoria lentă e numită *bloc*.
2) Politica de înlocuire: după o vreme cache-ul se va umple cu date, şi totuşi altele vor trebui aduse. Decizia despre care date trebuie scoase afară este foarte importantă pentru eficiență; ea este *politica de înlocuire* (replacement policy). Există o sumedenie de politici; iată-le doar numite pe unele; orice carte de sisteme de operare va descrie cele mai multe dintre ele: politica aleatoare (random), politica circulară (round robin), politica ``cel mai rar folosit'' (least frequently used), politica ``primul intrat - primul ieşit'' (first in, first out), politica ``cel mai demult folosit'' (least recently used), politica ``setul de lucru'' (working set), politica ``optimă'' (optimal), politica ceasului, politica celei de-a doua şanse (second-chance).
3) Politica de scriere: odată cu prezența unui cache, datele efectiv devin duplicate: există o copie în cache. Cînd se fac scrieri, care dintre copii trebuie modificată? Una sau amîndouă? În funcție de circumstanțe există varii răspunsuri la această întrebare.
4) Metoda de identificare: cînd se doreşte ceva din memoria lentă se indică *adresa* la care acel obiect se găseşte. Principiul transparenței (faptul că interfețele sunt identice) implică faptul ca în cache datele să fie căutate tot după această adresă; dar cum cache-ul este mic, adresele din memoria externă (= memoria lentă) nu reprezintă adrese şi în memoria rapidă. Cum găsesc datele? Răspunsul este dat de *metoda de identificare* şi strîns legat de politica de înlocuire, pentru că datele trebuie căutate acolo unde puteau fi aduse.
5) Timpul de viață al informației: dacă dintre copiile pe care le avem (una în cache şi alta în memorie) una se schimbă? Care este cea bună după aceea? Ce trebuie făcut cu cealaltă? Cu ce ocazie trebuie făcută schimbarea?

5.2.2. Cache-ul de disc

Orice sistem de operare modern (mai puțin MS-DOS) are un cache de disc. (Chiar şi pentru MS-DOS există `smartdrive` sau `ncache` de la Norton). Cache-ul de disc este probabil una din cele mai mari surse de eficiență într-un sistem de operare. Acesta se datoreşte faptului că diferența între timpul de acces la disc şi cel la memorie este uriaşă (timpul de acces al unei memorii este de circa 60-70 de nanosecunde, adică $60x10^{-9}$ s, iar timpul de acces al unui disc este de ordinul a 10 milisecunde, adică $10x10^{-3}$ s. Cache-ul de disc este o structură de date care conține un vector de blocuri de

mărime egală. Discul este la rîndul lui împărțit în blocuri de aceeaşi dimensiune. Cînd utilizatorul cere un octet de pe disc, blocul care conține acel octet este încărcat în cache, eventual scoțînd un alt bloc afară.

Din cele 5 puncte de vedere indicate anterior, un cache de disc are următoarele caracteristici:

1) Mărimea blocului: Blocuri mari (512 octeți - 8 Kb).
2) Politica de înlocuire: Politica de înlocuire cea mai frecventă este cea de excludere a datelor nefoşosite de cel mai mult timp (Least Recently Used).
3) Politica de scriere: Cache-urile de disc sunt în general ``write-back''. Asta înseamnă că atunci cînd se *scrie* pe disc, modificările sunt făcute doar în cache. Ele sunt mutate pe disc doar cînd blocul respectiv este dat afară, sau cînd acest lucru este cerut explicit de utilizator.
4) Metoda de identificare: Pentru a găsi un bloc în cache se folosesc algoritmi de *hash*, care sunt foarte eficienți (orice carte elementară de algoritmi descrie hash-urile).
5) Timpul de viață al informației din cache: Pentru a preveni catastrofele, sistemele de operare ``golesc'' (scriu toate blocurile modificate) din cache pe disc periodic (de ex. la 30 de secunde).

5.2.3. Cache-ul microprocesorului

Un microprocesor la 200 de Megahertzi (un Pentium pro, de pildă) are un ciclu de instrucțiune de$1/(200x10^6) = 5$ nanosecunde. O instrucțiune poate dura un număr variabil de cicluri, între 1 şi cîteva zeci. Executarea unei instrucțiuni înseamnă: citirea ei din memorie, decodificarea, executarea, memorarea rezultatelor. Dacă accesul la memorie durează 60 de nanosecunde atunci la fiecare citire procesorul trebuie să piardă 12 cicluri! Din cauza asta între microprocesor şi memoria RAM principală se pune un cache construit din memorie rapidă, cu timp de acces de 5-10 nanosecunde.

Cîteodată designerii pun chiar mai mult decît atît: două nivele de cache între procesor şi RAM: un nivel ceva mai lent, dar mai mare (pentru un PC între 64Kb şi 512Kb de obicei), şi un cache construit chiar în microprocesor, de ordinul a 1-10Kb, mult mai rapid.
Aceste cache-uri se implementează folosind hardware specializat.

1) Mărimea blocului: blocurile sunt mici 1 - 16 octeți.
2) Politica de înlocuire si **4)** Metoda de identificare:

Există două clase mari de cache-uri de microprocesor, şi una intermediară. Ele diferă prin locurile din cache în care un octet din memoria externă poate fi plasat. Cele două mari varietăți sunt: cache-ul cu adresare directă, în care locul fiecărui octet este unul şi precis calculat, şi cache-ul asociativ, în care un octet din memoria externă poate fi plasat în orice loc din cache.

5.2.4. Cache-ul cu adresare directă (direct mapped)

De obicei chiar structura adresei este folosită la căutare. Figura 5.8 arată cum este plasat un anume bloc în cache: biţii de la sfîrşitul adresei blocului dau şi posibila poziţie a blocului în cache. Biţii din începutul adresei blocului constituie verificarea

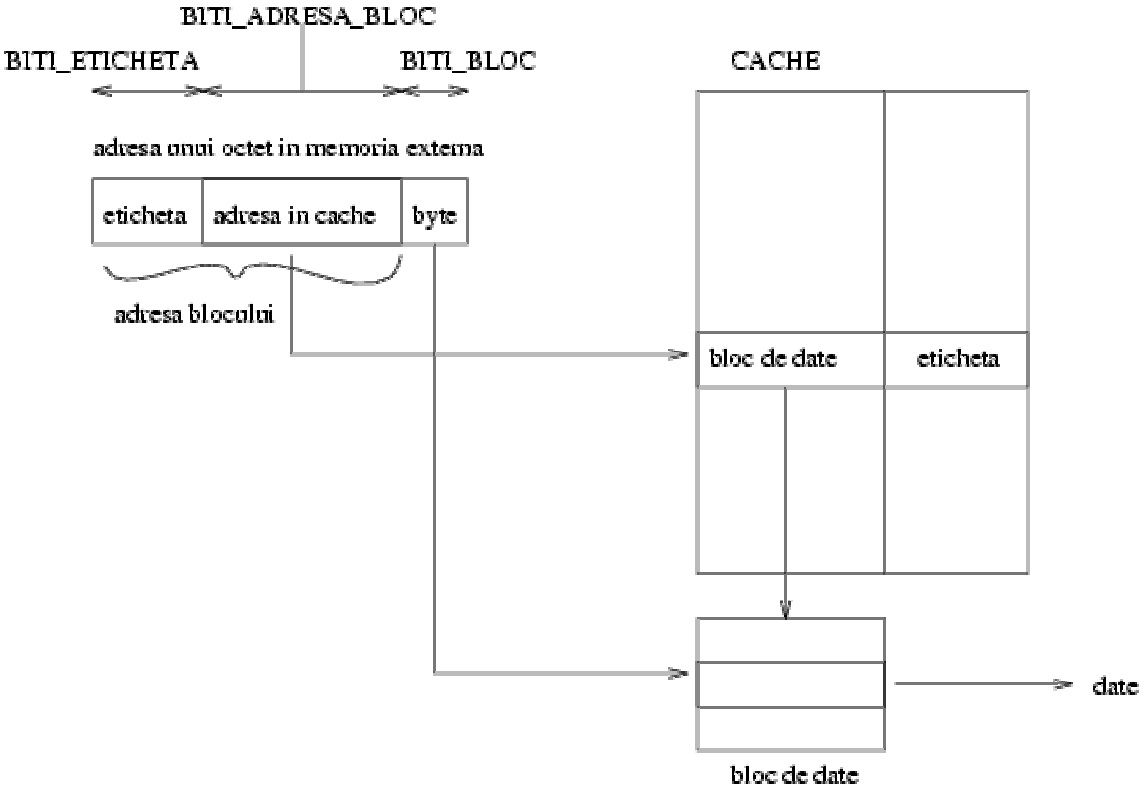


Figura 5.8. Cache-ul cu adresare directă

dacă blocul este cel aflat în cache (mai multe blocuri candidează pentru aceeaşi poziţie; cel care se află înauntru este indicat prin *etichetă* (tag)).

În fine, ultimii biţi din adresă indică poziţia octetului în blocul de date.

Marele avantaj al schemei directe este că dată fiind adresa, poziţia în cache a blocului este unic determinată, şi nu trebuie făcută nici o căutare. Politica de înlocuire nu există din acelaşi motiv: nu poţi alege în ce loc să aduci un bloc. Din cauza asta funcţia de căutare şi cea de înlocuire sunt identice.

5.2.5. Cache-ul cu adresare asociativă (fully associative)

Cache-ul cu adresare asociativă se bazează pe un dispozitiv hardware foarte simpatic, care se numeşte *memorie asociativă* (din cauza prezenţei ei îşi capătă cache-ul numele).

O memorie obişnuită oferă două operaţii: (a) dîndu-se o adresă, citeşte conţinutul şi (b) dindu-se o adresă şi o valoare scrie această valoare acolo.

Pe lîngă aceste operaţii o memorie asociativă mai oferă încă una: dîndu-se o valoare, poate spune la care adresă se găseşte ea. O memorie asociativă nu este tehnologic greu de construit, însă este un dispozitiv relativ costisitor.

Un cache asociativ foloseşte o memorie asociativă pentru a memora *adresele externe* ale blocurilor care corespund fiecărui bloc din cache.

Un bloc poate acum ocupa orice poziție în cache; cînd este căutat memoria asociativă spune unde se află.

Politica de înlocuire va fi însă ceva mai complicată, oricare din schemele înşirate fiind un candidat.

5.2.6. Cache-ul parțial asociativ (set-associative)

Putem să ne imaginăm un cache parțial asociativ ca o colecție de mai multe cache-uri directe care lucrează în paralel. Fie k numărul de astfel de cache-uri directe. (un astfel de cache se numeşte ``associative on k ways" -- asociativ pe k direcții).

Ideea este simplă: cînd caut o adresă folosesc adresare directă în *toate* cele k cache-uri directe simultan. Dacă blocul se găseşte într-unul am rezolvat problema. Daca nu, aleg unul dintre ele pentru înlocuire. Numele este de ``parțial asociativ", pentru că plasamentul în cele k blocuri posibile este oricare, ca la un cache asociativ.

Să revenim la discuția privind cache-urile microprocesoarelor.

[3)] Politica de scriere, **5)** Timpul de viață al informației din cache: Dacă mai multe microprocesoare sunt legate la aceeaşi memorie, există riscul ca fiecare să facă modificări în propriul cache, obținînd astfel rezultate eronate, aşa cum arată şi figura 5.9. Din cauza asta cache-ul se face adesea ``write-through": toate modificările se fac simultan în memorie şi în cache. Cache-urile monitorizează modificările făcute în memorie de celelalte cache-uri şi invalidează copiile datelor pe care le posedă şi care au fost modificate. (Un astfel de cache se numeşte ``snooping cache": cache care trage cu urechea, să vadă dacă altcineva nu modifică memoria externă).

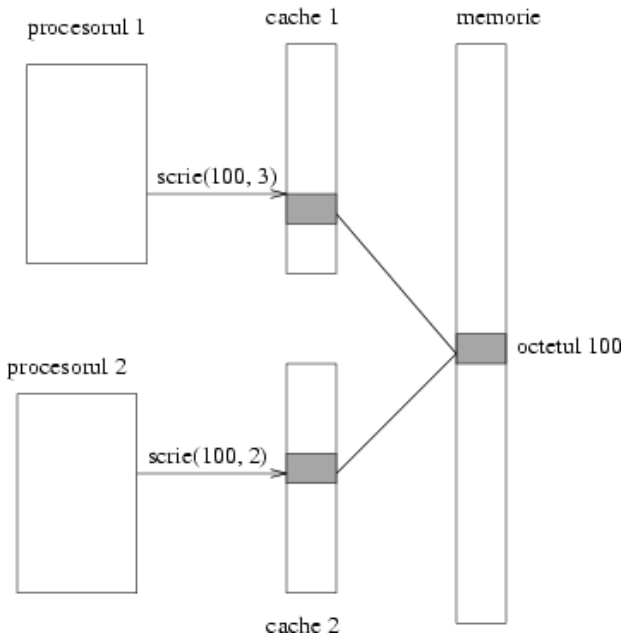


Figura 5.9. Acces prin două cache-uri

5.3. Gruparea memoriilor

Aşa cum s-a arătat, una din caracteristicile importante ale memoriei este reprezentată de capacitatea acesteia. Capacitatea reflectă cantitatea de informaţie ce se poate stoca într-o memorie şi este strâns legată de structura acesteia. Gruparea memoriilor reprezintă metoda de creştere a capacităţii de stocare prin conectarea împreună a mai multor memorii de capacitate mai mică. Pentru a vedea care sunt metodele de grupare a memoriilor vom studia mai întâi semnalele unei memorii.

O memorie are nevoie de mai multe semnale de intrare şi poate furniza mai multe semnale de ieşire. Aşa cum s-a arătat în paragraful 5.1, o memorie este alcătuită din mai multe celule de o anumită dimensiune (un anumit număr de biţi) care sunt identificate prin adresă. Din acest punct de vedere, o memorie necesită un anumit număr de adrese cu ajutorul cărora se identifică fiecare celulă de memorie în parte. Semnalele de adresă sunt semnale binare (pot avea valoarea zero sau unu), fiecare celulă de memorie fiind identificată de o combinaţie unică de pe magistrala de adrese. Astfel, dacă o memorie are, spre exemplu, opt celule de memorie, atunci sunt necesare opt combinaţii distincte de cifre zero sau unu. Cele opt combinaţii distincte pot fi obţinute cu ajutorul a $\log_2 8 = 3$ cifre binare. Altfel spus, cu ajutorul a trei cifre binare se pot obţine $2^3 = 8$ combinaţii distincte: 000, 001, 010, 011, 100, 101, 110 şi 111.

Celula de memorie selectată permite citirea sau scrierea *simultană* a tuturor biţilor pe care aceasta îi conţine. Din acest motiv, o memorie necesită pe lângă liniile de adresă şi un anumit număr de linii de date, numărul liniilor de date fiind egal cu numărul de biţi ai celulei de memorie. Liniile de date trebuie să fie bidirecţionale (ca sa permită citirea sau scrierea biţilor celulei de memorie) cu trei stări (să poată trece în starea de înaltă impedanţă) pentru a permite conectarea în paralel a mai multor memorii. Este evident că în cazul memoriilor ROM liniile de date nu sunt bidirecţionale din cauză că aceste memorii nu pot fi decât citite dar trebuie, de asemenea să aibă starea de înaltă impedanţă pentru a putea fi conectate la magistrala de date a sistemului.

Cea de a treia categorie de semnale necesare unei memorii sunt semnalele de comandă. Semnalele de comandă principale ale unei memoriei sunt:

- semnalul de comandă al selectiei circuitului (Chip Select – notat CS) care este un semnal de intrare în memorie şi care determină starea magistralei de date (acest semnal nu poate lipsi la nici un tip de memorie):
  - CS = 1, pe magistrala de date circulă semnale, fiecare linie a magistralei putând avea un semnal care este în starea unu sau zero logic (în memorie se poate scrie sau din memorie se poate citi);
  - CS = 0, pe magistrala de date nu circulă semnale, aceasta fiind în starea de înaltă impedanţă;
- semnalul de comandă al citirii din memorie sau scrierii în memorie ($\mathrm{Re}ad/\overline{Write}$ - notat $R/\overline{W}$) care este un semnal ce determină direcţia magistralei de date (acest semnal lipseste la memoriile ROM):
  - $R/\overline{W} = 1$, din memorie se citeste, sensul magistralei de date fiind din memorie spre exterior; la un moment dat se citesc biţii celulei

de memorie, selectate de combinația aflată pe magistrala de adrese;

- $R/\overline{W}=0$, în memorie se scrie, sensul magistralei de date fiind din exterior spre memorie; la un moment dat se scriu biții celulei de memorie, selectate de combinația aflată pe magistrala de adrese;

Pe lîngă aceste semnale, mai există o serie de semnale de comandă diverse, cum ar fi semnalele de reâmprospătare, care nu sunt implicate în funcționarea de bază a memoriei.

Memoria, ca matrice, este definită de declarația:

memoria: array (0…p) of byte

unde ***p*** reprezintă numărul de celule de memorie cu dimensiunea de un octet (byte). O astfel de memorie are nevoie de $\log_2 n$ linii de adrese şi opt linii de date.

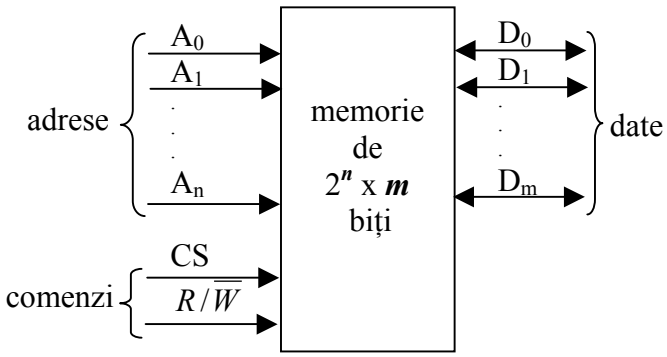


Figura 5.10. Schema bloc a memoriei.

Bineînțeles că celula de memorie poate avea şi dimensiunea de un cuvânt (word) sau pointer sau dimensiuni mai mici cum ar fi 1 bit sau 4 biți.

În continuare vom considera memorii generice care au adrese, date şi semnalele de comandă: CS şi $R/\overline{W}$, aşa cum se arată în schema bloc a unei memorii generice din figura 5.10.

Capacitatea memoriei poate fi exprimată de către o relație similară cu relația (5.1).

capacitatea = $2^{\text{numărul de linii de adresă } (n+1)}$ x numărul de linii de date ($m+1$) (5.1)

De exemplu, capacitatea unei memorii cu zece linii de adresă şi opt linii de date va fi:

$$C = 2^{10} \text{ x } 8 \text{ biți} = 1024 \text{ x } 8 \text{ biți} = 1\text{k octet} = 1\text{ko}$$

Capacitatea exprimată în forma dată de ecuația 5.1 se numeşte capacitate structurală deoarece permite determinarea structurii unei memorii. Pentru exemplul anterior avem: C = 1024 x 1 octet, care ne permite să aflăm structura memoriei: 10 linii de adresă ($\log_2 1024 = 10$), care pot selecta în mod univoc 1024 ($2^{10} = 1024$) celule de memorie care au câte 8 biți (magistrala de date este de un octet).

Capacitatea se poate exprima şi prin valoarea ei absolută, rezultatul înmulțirii numărului de celule de memorie cu dimensiunea acestora, obținându-se numărul de

celule *elementare* de memorie cu dimensiunea de un bit (de exemplu 1ko), valoare care însă nu ne mai dă informații asupra structurii. O memorie de 1ko poate fi: o memorie de 1024 x 1 octet sau o memorie de 2048 x 4 biți.

Gruparea memoriilor poate face în trei moduri:

- gruparea în scopul creşterii numărului de linii de date ale memoriei, numărul de linii de adresă rămânând neschimbat; creşterea numărului de linii de date a memoriei înseamnă de fapt creşterea numărului de biți ai celulei de memorie (numărul de biți citiți sau scrişi din/în memoria respectivă) ai memoriei rezultate. Evident că mărirea capacității celulei de memorie va determina creşterea corespunzătoare a capacității totale a memoriei;
- gruparea în scopul creşterii numărului de linii de adresă a memoriei, numărul liniilor de date rămânând nemodificat; creşterea numărului de linii de adresă înseamnă de fapt cresterea numărului de celule de memorie ai memoriei rezultate, capacitatea celulei (numărul de linii de date) rămânând neschimbat; şi în acest se produce modificarea corespunzătoare a capacității memoriei;
- gruparea mixtă în care se creşte atât numărul de linii de adresă cât şi numărul liniilor de date.

5.3.1. Creşterea capacității memoriei prin creşterea numărului de linii de date

Creşterea numărului de linii de date este necesară, spre exemplu, atunci când dispunem de memorii cu opt linii de date (un octet) şi dorim să le conectăm la un sistem care are o magistrală de date cu 16 linii (un cuvânt – word).

Vom exemplifica modul de grupare pe acest exemplu. Presupunem că avem nevoie de o memorie de 8kw (8 kilo word – 8 kilo cuvinte) şi nu dispunem decât de memorii de 8kb (8 kilo byte – 8 kilo octeți). Se observă faptul că între memoriile disponibile şi cele necesare nu diferă decât numărul de linii de date (necesar 16, existent 8), numărul de linii de adresă fiind acelaşi (8192 de celule înseamnă $\log_2 8192 = 13$ linii de adresă: A_0 la A_{12}). Vom determina mai întâi cîte memorii sunt necesare. Capacitatea absolută a memoriei ce trebuie obținută este:

$$C = 8192 \text{ x } 16 \text{ biți} = 131\,072 \text{ biți}.$$

Capacitatea memoriei existente este:

$$C = 8192 \text{ x } 8 \text{ biți} = 65\,536 \text{ biți}.$$

Rezultă deci că pentru a obține o memorie de 8 kword avem nevoie de 131072/65536 = 2 memorii de 8 kbytes.

Modul de conectare a celor două memorii de 8 kb pentru obținerea unei memorii de 8 kiloword este prezentat în figura 5.11.

Se observă că pentru gruparea memoriilor în vederea extinderii numărului de linii de date, liniile de adresă şi liniile de comandă se leagă în paralel iar liniile de date se pun împreună formând magistrala de date a memoriei rezultate.

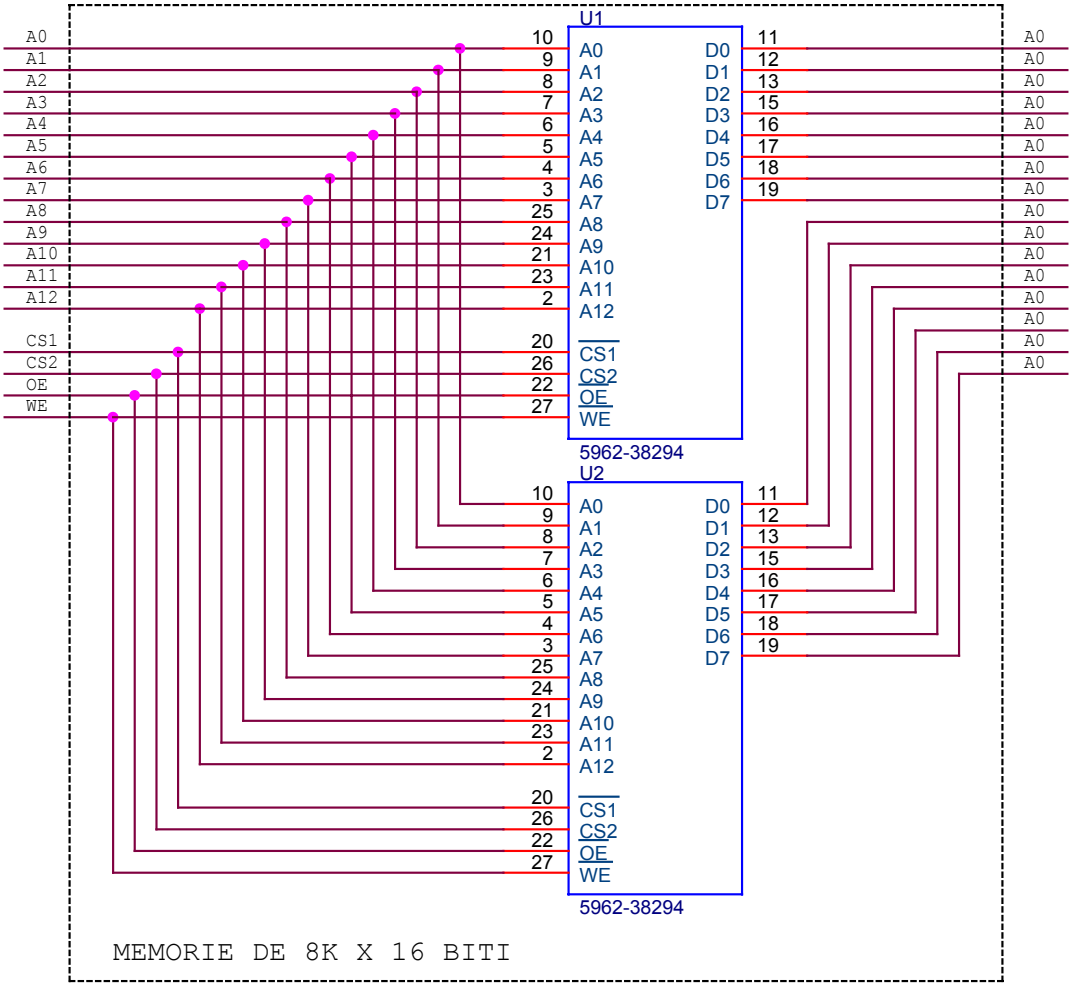


Figura 5.11. Gruparea memoriilor pentru creşterea numărului liniilor de date.

5.3.2. Gruparea memoriilor pentru creşterea numărului de linii de adresă

Creşterea numărului de linii de adresă este necesară pentru creşterea capacităţii unei memorii fără modificarea dimensiunii celulei de memorie.

Pentru a explica principiul vom considera un exemplu simplu. Să presupunem că dorim să obţinem o memorie de 8 x 1 bit cu ajutorul unor memorii de 4 x 1 bit. Atât memoria pe care dorim s-o obţinem cât şi memoriile disponibile au celule de aceeaşi dimensiune (un bit). Ceea ce se doreşte este dreşterea numărului acestor celule. Memoriile disponibile, cele de 4 x 1 bit, necesită două linii de adresă pentru selectarea uneia din cele patru celule iar memoria care se doreşte a fi obţinută, cea de 8 x 1 bit are nevoie de 3 linii de adresă. Rezultă că numărul de linii de adresă trebuie crescut de la două linii la trei.

Vom analiza acum modul de adresare al celulelor de memorie. Fiecare celulă a memoriei este adresată în mod univoc de către o combinație binară de pe magistrala de adrese aşa cum se arată în tabelul 5.3. Din acest tabel se observă faptul că cele patru combinații ale liniilor de adresă A_1A_0 se repetă de două ori la memoria de 8 x 1 bit, o dată pentru $A_2 = 0$ şi o dată pentru $A_2 = 1$. Rezultă că este posibil să realizăm memoria de 8 x 1 bit cu ajutorul a două memorii de 4 x 1 bit este necesar sa selectam una din memoriile de 4 x 1 bit când $A_2 = 0$ şi să selectăm cealalta memorie de 4 x 1 bit când $A_2 = 1$. Rezultă că semnalele de selacție a memoriilor de 4 x 1 bit vor fi folosite pentru crearea liniei de adresă A2 şi va trebui sa creem un nou semnal de selecție pentru memoria rezultata. Vom exemplifica pe acelaşi tip de memorie de 8k x 1 octet. Prin gruparea acestora în scopul creşterii numărului de linii de adresă (creşterea numărului de celule de opt biți) se va crea o noua linie de adresă (A_{13}) memoria obținută având 16k x 1 octet.

TABELUL 5.3.

memorie 4x1bit		memorie 8 x 1bit		
A_1	A_0	A_2	A_1	A_0
0	0	0	0	0
0	1	0	0	1
1	0	0	1	0
1	1	0	1	1
		1	0	0
		1	0	1
		1	1	0
		1	1	1

În acest caz, liniile de date se cuplează în paralel deoarece dimensiunea celulei de memorie rămâne nemodificată (figura 5.12). Memoria inactivă la un moment dat are

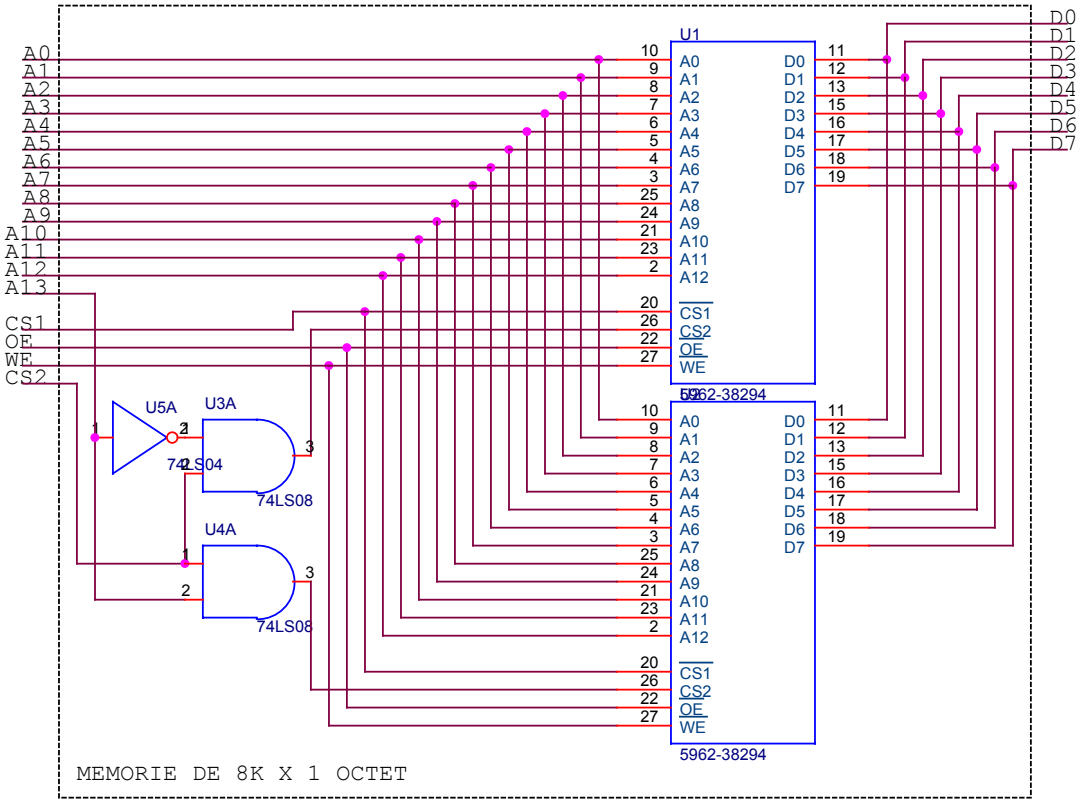


Figura 5.12. Gruparea memoriilor prin extinderea liniilor de adresă.

liniile de date în starea de înaltă impedanță (prin comanda selecției ciruitului respectiv) şi în acest fel liniile de date ale celor două memorii pot fi cuplate împreună.

OBSERVAȚIE: din cele două exemple prezentate anterior rezultă valoarea capacității memoriei finale este multiplu întreg de valoarea capacitîții memoriei finale.

5.3.3. Gruparea mixtă

Gruparea mixtă constă în creşterea atât a numărului de linii de date cât şi a creşterii numărului de linii de adresă ceea ce inseamnă creşterea dimensiunii celulei de memoriei şi a numărului de celule.

Pentru a realiza o astfel de grupare se realizează mai întâi gruparea memoriilor în vederea cresterii numărului de linii de date după care memoriile obținute se grupează prin cresterea numărului de linii de adresă, obținându-se în final memoria dorită.

Vom exemplifica acest lucru prin realizarea unei memorii de 2k octeți cu ajutorul unor memorii de 1k x 4 biți.

Capacitatea absolută a memoriei rezultate este:

$$C = 2 \text{ x } 1024 \text{ x } 8 \text{ biți} = 16384 \text{ biți}$$

Această capacitate trebuie să fie multiplu întreg de capacitatea memoriilor inițiale. Memoria inițiala are capacitatea absolută:

$$C = 1024 \text{ x } 4 \text{ biți} = 4096 \text{ biți.}$$

Rezultă că sunt necesare 16384 : 4096 = 4 memorii de 1k x 4 biți. Aceste patru memorii se vor grupa astfel: mai întâi memoriile se grupează două câte două prin extinderea numărului de linii de date, obținânduse două memorii cu dimensiunea de 1k x 8 biți. Cele două memorii obținute se grupează prin extinderea numărului de linii de adresă, rezultând memoria finală de 2k octeți.

5.4. Adresarea memoriilor

Într-un sistem de calcul, principalul modul master care generează adrese este unitatea centrală. Numărul total de adrese pe care-l poate genera unitatea centrală reprezintă *spațiul maxim de adresare*. Acest spațiu se împarte, conform semnalelor de comandă generate de unitatea centrală, între memorie şi porturi. Există mai multe moduri de adresare a acestora în funcție de utilizarea spațiului maxim de adresare. Adresarea memoriei (sau a porturilor) poate fi *absolută* dacă la formarea adresei se folosesc toți biții de adresă sau relativă (care se mai numeşte şi *redundantă*) dacă se folosesc numai o parte din biții de adresă.

5.4.1. Adresarea absolută

La adresarea absolută se folosesc toți biții de adresă de pe magistrala de adrese pentru selectarea unei locații de memorie (sau a unui port).

valoare în zecimal	adrese pentru selecția memoriei de 1k						adrese pentru selecția unei celule din memoria de 1k				
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	…	A_1	A_0
0	0	0	0	0	0	0	x	x	…	x	x
1	0	0	0	0	0	1	x	x	…	x	x
2	0	0	0	0	1	0	x	x	…	x	x
.	.										
.	.										
.	.										
63	1	1	1	1	1	1	x	x	…	x	x

x x … x x -> orice combinație între 00…00 şi 11 … 11

Figura 5.13. Posibilitățile de adresare absolută a unei memorii de 1k.

Vom explica adresarea absolută printr-un exemplu. Considerăm că într-un sistem magistrala de adrese are 16 linii (A_0, A_1, … , A_{15}) deci spațiul maxim de adresare este de 64 k (2^{16} = 65536). Pe acestă magistrală dorim să conectăm o memorie de 1k (dimensiunea datelor nu are importanță aici). Pentru adresarea celor 1024 de celule de memorie din memoria de 1k avem nevoie de 10 linii de adresă (A_0, A_1, … , A_9) restul de 6 linii de adresă (A_{15}, A_{14}, … , A_{10}) fiind folosiți la selecția circuitului. Astfel adresele A_0, A_1, … , A_9 se leagă la liniile de adresă ale memoriei iar adresele A_{10}, A_{11}, … , A_{15} se leagă printr-un circuit de selecție la intrarea CS a memoriei. În acest fel memoria poate fi aşezată la una din cele 64 de adrese posibile din memorie (figura 5.13).

Prin adresarea absolută, o locație de memorie se găseşte la o singură adresă de memorie.

5.4.2. Adresarea relativă (redundantă)

În cazul adresării redundante, adresele (sau o parte dintre acestea) nu sunt folosite la selecția memoriei. În acest fel o celulă de memorie se poate găsi la mai multe adrese diferite.

Reluând exemplul din paragraful anterior, dacă se leagă semnalul de selecție al memoriei de 1k la valoarea unu logic (memorie permanent selectată) şi în acest fel se face abstracție de adresele A_{15}, A_{14}, … , A_{10}, adresarea unei locații din această memorie este redundantă.

Astfel, celula de memorie de la adresa 0 se va găsi la 64 de adrese distincte: 0, 400h, 800h, C00h şi aşa mai departe, pâna la adresa FFFFh. Acest lucru se întâmplă deoarece numai adresele A_0, A_1, … , A_9 participa la selecția celulei, adresele fiind ignorate.

CAPITOLUL 6

PORTURI (INTERFEȚE)

6.1. Prezentare generală

Porturile sunt module destinate conectării sistemului de calcul cu dispozitivele externe care sunt numite generic dispozitive ***periferice***. La porturi se conectează atât dispozitivele generale de intrare/ieşire (tastatură, monitor) ale calculatorului cât şi cele specifice. De asemenea două porturi de aceeaşi natură se pot conecta între ele în scopul transmiterii informației de la un sistem de calcul la altul. Porturile sunt în general circuite de viteze mică, ele fiind conectate la magistralele de viteză mai redusă ale calculatorului. O parte din porturile sistemului de calcul au destinații specifice şi poartă denumirea de ***controlere***. Dintre acestea mai importante sunt: controlerul de tastatură, controlerul video, controlerul discului dur (hard disc), controlerul discului flexibil (floppy disc) şi controlerul de rețea. Alte porturi au destinații generale, la aceste porturi putând fi conectate diferite dispozitive periferice; dintre porturile de uz general mai importante fiind portul serial, portul paralel, portul SCSI (Small Computer System Interface) şi portul USB (Universal Serial Bus). Datorită faptului că, în general, un port trebuie să realizeze atât conversia semnalelor dispozitivelor periferice la semnale compatibile cu semnalele sistemului de calcul, cât şi invers, asigurând circulația informației în ambele sensuri, acestea se mai numesc şi ***interfețe***. Prin interfață se înțelege un ansamblu format din echipamente şi programe destinat cuplării între două sisteme cu caracteristici diferite. Dispozitivele periferice se conectează la porturi prin intermediul unor conectori, un exemplu fiind prezentat în figura 6.1.

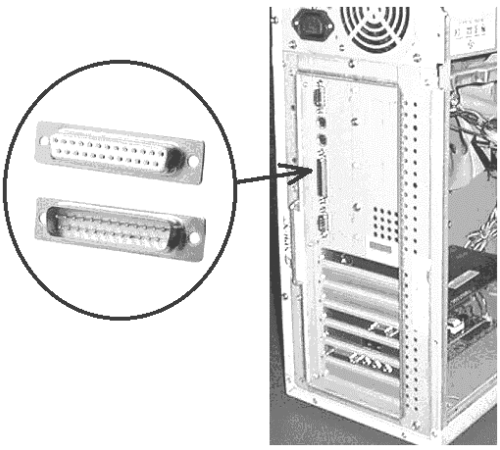


Fig. 6.1. Conectorii pentru porturi în cazul unui calculator personal

Fiecare port are alocată o adresă (un număr) pe care unitatea centrală o foloseşte pentru identificarea portului respectiv în schimbul de date cu acesta. Este foarte important ca toate porturile dintr-un calculator să aibă adrese diferite. Dacă două porturi au aceeaşi adresă, atunci apare un conflict din cauză că unitatea centrală nu va fi capabilă să distingă de la care port sosesc informațiile.

În general activitatea de atribuire a adreselor tuturor porturilor dintr-un sistem de calcul se face în mod automat de către ***sistemul de operare***. Uneori, dacă numărul

perifericelor este mare sau din generații diferite, se poate întâmpla ca sistemul de operare să nu poată realiza alocarea corectă a adreselor perifericelor. În acest caz alocarea adreselor se poate face manual de către operator.

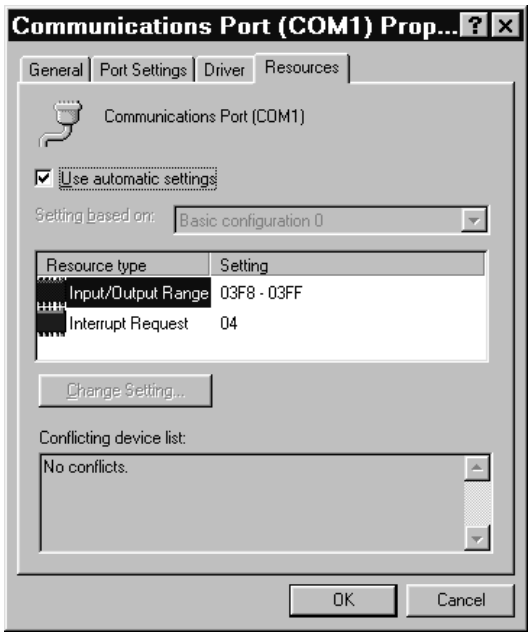


Fig. 6.2. Alocarea adresei şi întreruperii la portul serial COM1

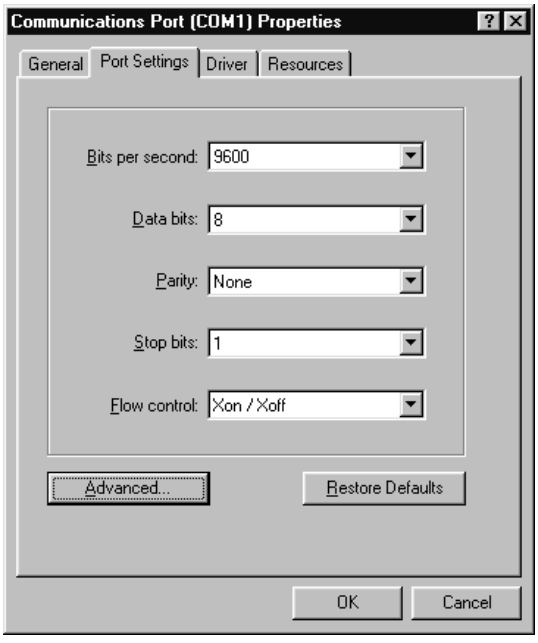


Fig. 6.3. Parametrii portului serial

Porturile fiind în general componente lente ale sistemului de calcul, este neeconomic ca unitatea centrală să lucreze la un moment dat numai cu un port. Modul de lucru cu un port se desfăşoară simplificat în felul următor: unitatea centrală trimite o comandă la un port după care continuă rezolvarea altor sarcini. În momentul în care portul este capabil să răspundă solicitării, întrerupe activitatea curentă a unității centrale, primeşte o nouă comandă, după care unitatea centrală nu se mai ocupă de port până la o nouă întrerupere. În felul acesta oricât de lent este un port el nu scade viteza de lucru a unității centrale. Acest mecanism necesită, pe lângă alocarea unei adrese şi alocarea unei ***întreruperi*** (un număr) fiecărui port în aşa fel încât în momentul lansării unei cereri de întrerupere, unitatea centrală a calculatorului să poată determina care este portul care a cerut întreruperea. Evident că şi în cazul întreruperii este bine ca fiecare port să dispună de o întrerupere separată. Deşi această condiție nu este la fel de restrictivă ca cea în cazul adresei, în unele cazuri mai multe porturi putând avea alocată aceeaşi întrerupere, în marea majoritate a cazurilor numărul întreruperii trebuie să fie diferit de la un port la altul.

Din cele arătate rezultă că un port necesită alocarea din partea sistemului de operare sau a programatorului a două numere: adresa şi numărul întreruperii, fapt ce-l individualizează față de celelalte porturi din sistem. În figura 6.2 este prezentat modul în care sistemul de operare Windows'95 oferă informații despre setările unui port. Din această figură se vede că portul de comunicații COM1 are alocate adresele de la 03F8h la 03FFh şi întreruperea 04.

Portul serial al calculatorului necesită o atenţie specială din cauză că el este în general portul de comunicaţii. Prin intermediul acestui port se pot transmite date la distanţă în mod ***serial asincron***. Transmisia serială este metoda cea mai ieftină de a schimba date între două echipamente numerice aflate la distanţă. Deşi transmisiile seriale nu sunt transmisii de date de viteză mare ele prezintă avantajul că datele sunt transmise pe un singur canal de comunicaţie şi deci nu sunt necesare cheltuieli mari. Modul serial de transmisie presupune ca la un moment dat să se transmită un singur bit. Asta înseamnă că biţii unui octet vor fi transmişi la opt intervale de timp distincte. În cazul transmisiei seriale este necesar ca atât echipamentul care transmite datele (emiţătorul) cât şi echipamentul care primeşte datele (receptorul) să aibă aceeaşi ***parametrii ai transmisiei***. Pentru transmisia serială asincronă parametrii transmisiei sunt: viteza de transmisie care se măsoară în biţi pe secundă (bps), numărul de biţi transmişi într-un cadru, paritatea, numărul de biţi de stop şi protocolul de control al fluxului de date. În figura 6.3 este prezentat un exemplu de stabilire ai parametrilor transmisiei. Standardul adoptat pentru interfaţa seriala este standardul RS 232.

Porturile sunt în general dispozitive programabile. Asta înseamnă că ele acceptă prin înscrierea unor cuvinte de comandă în port. Din acest motiv, pentru funcţionarea corectă a unui port nu sunt suficiente numai alocarea adresei şi a întreruperii ci este necesară şi programarea portului. Pentru simplificarea programării şi utilizării portului se folosesc nişte programe specifice numite ***drivere***. Aceste programe sunt încărcate în memorie de către sistemul de operare şi ele constituie o interfaţă între programele utilizatorului şi port. În figura 6.4 este reprezentat schematic modul de funcţionare a unui driver. Programele driver sunt furnizate de către producătorii de echipamente periferice şi asigură funcţionarea optimă a acestora. Din acest motiv utilizatorul unui sistem de calcul nu are decât sarcina de a obţine de la producătorii de echipamente periferice a versiunilor noi ale driverelor şi să le instaleze pe calculator.

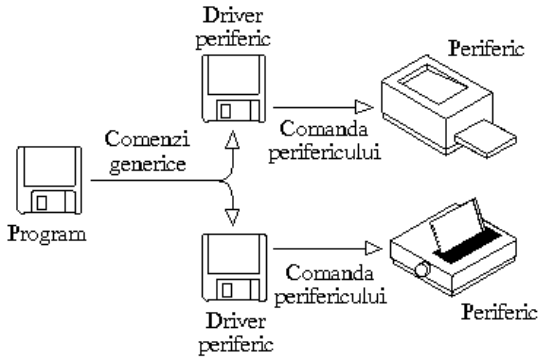


Fig. 6.4. Funcţionarea unui program driver pentru un port

Noua tehnologie de realizare a porturilor inteligente s-a extins tot mai mult. Cu ajutorul acestei tehnologii, utilizatorul este degrevat de sarcina de a mai configura porturile sistemului, acestea fiind recunoscute automat şi programate corespunzător de către programele cu care este înzestrat sistemul de calcul. Această tehnologie numită PnP (Plug and Play) permite utilizatorului să realizeze extinderi ale sistemului de calcul simplu şi comod, aşa cum arată şi numele tehnologiei, prin simpla montare a componentei noi în sistemul de calcul, fără a mai fi necesare alte operaţii suplimentare.

În continuare vor fi prezentate porturile dezvoltate de firma Intel pentru sistemul de calcul prevăzut cu unitatea centrală I8086. Datorită faptului că aceste dispozitive au fost larg răspândite ele au devenit un standard "de facto" pentru circuitele dezvoltate ulterior.